UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Predicción de deforestación amazónica con redes neuronales (1819_1684_COISTITE)

Luis de Juan del Villar

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Predicción de deforestación amazónica con redes neuronales (1819_1684_COISTITE)

Autor: Luis de Juan del Villar
Tutor: David Domínguez

julio 2020

Luis de Juan del Villar
Predicción de deforestación amazónica con redes neuronales (1819_1684_COISTITE)

Luis de Juan del Villar

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi abuela, que en paz descanse

“Birds inspired us to fly, burdock plants inspired velcro, and countless more inventions were inspired by nature.
It seems only logical, then, to look at the brain’s architecture for inspiration on how to build an intelligent machine.”

Aurelien Geron

Prefacio

La Amazonia es el bosque tropical más extenso del mundo, situado en Sudamérica. Su superficie original oscila entre 5,5M y 6,7M km^2. La región amazónica pertenece a nueve países, de los cuales Brasil abarca la mayor extensión. La selva amazónica destaca por la gran biodiversidad, ofreciendo una fauna y flora muy ricas en cantidad y variedad.

Dado el tamaño de esta selva, produce una gran cantidad de oxígeno, contribuyendo a que dispongamos de un aire más limpio en el planeta. Sobra decir que se trata de un excepcional ecosistema que tenemos que cuidar para que las generaciones venideras puedan disfrutar de los muchos beneficios que ofrece, así como nosotros lo hacemos con esta maravilla que nos ha sido regalada.

Actualmente no se están poniendo los medios necesarios para mantenerlo. Los incendios, y la tala masiva de árboles están acabando con la Amazonia. En los últimos años se está elevando la deforestación y esto puede llevarnos a una situación irreversible. Además, dada la coyuntura socioeconómica y política actual, las previsiones son pesimistas.

En este Trabajo de Fin de Grado trato de sensibilizar al respecto, estimando predicciones de los niveles de deforestación que se van a producir a lo largo de los los próximos años. Para ello, me baso en un histórico de datos de deforestación en los últimos años y entreno una red neuronal que combina un perceptrón con una red LSTM.

Luis de Juan del Villar

Agradecimientos

Me gustaría agradecer este TFG a mis padres y a mi pareja, que siempre me han apoyado, en las buenas y en las malas. Cuando he pasado por momentos difíciles, me han comprendido, consolado, motivado y animado a creer en mí mismo. Se han asegurado de poner los medios que les ha sido posible para que yo pueda centrarme al máximo en mi trabajo. Me han dado las fuerzas para sentir que soy capaz de todo. Sin ellos, no habría llegado tan lejos.

Quiero mencionar a dos compañeros de la carrera: Miguel Sánchez Durán, quien ha sido mi compañero de prácticas en la mayoría de asignaturas, y Dionisio Pérez Alvear, con quien he vivido mi primer año y medio de experiencia laboral. Con ambos he compartido noches enteras de programación, tabaco, comida poco sana y, sobre todo, mucha satisfacción por el trabajo bien hecho. Ha sido muy enriquecedor formar equipo con gente trabajadora y competente. Además, puedo prometer y prometo que he disfrutado de un buen ambiente de trabajo con personas en la que se puede confiar y con un corazón que no les cabe en el pecho.

Merece ser destacada la figura de Michele Grazioli, una persona a la que admiro tanto humana como profesionalmente. He tenido la oportunidad de trabajar con él y para mí ha sido un mentor en el campo de la inteligencia artificial.

Por último, quiero destacar a mis amigos, que en su mayoría no tienen ni idea de lo que hago. Quizás por eso, me admiran y se piensan que soy un genio. Su confianza en mí me hace crecer y me motiva para encarar nuevos retos, cogiendo el toro por los cuernos y confiando en que soy capaz de todo; no hay límite.

Resumen

En este TFG se pretende hacer uso de la tecnología para realizar un análisis del impacto humano en el medioambiente. En primer lugar, se pretende recalcar la importancia de la Amazonia en el planeta y la deforestación desmedida a la que se está viendo sometida, que puede ser analizada a partir de los datos y predicha por medio de la inteligencia artificial. Seguidamente, se ofrece una detallada explicación del funcionamiento de las redes neuronales, fundamentales en el modelo diseñado. Posteriormente, se detalla el problema planteado, que consiste en predecir la deforestación en la Amazonia en los próximos años. Se expone el preprocesamiento de los datos, el modelo diseñado, detalles de implementación y, por último, los resultados, conclusiones y trabajo futuro.

Concretamente, el modelo trata de predecir, mediante redes neuronales, la deforestación de la Amazonia en los próximos años. Se dispone de un conjunto de datos que contiene datos de 760 municipios, incluyendo datos fijos (latitud, longitud, área, hidrografía, zona no forestal, etc) y datos anuales de deforestación en lo últimos 20 años. Por ello, el modelo que se ha desarrollado combina redes neuronales densas para variables fijas y redes neuronales recurrentes LSTM para variables temporales.

Es un problema complejo, ya que se trata de un regresor, que generalmente supone más dificultad que un clasificador. Las variables de las que se dispone son fijas y temporales, lo que la añade una complicación mayor. Además el volumen del conjunto de datos no es el deseable.

Se han realizado muchas iteraciones probando distintas configuraciones del modelo, ajustando hiperparámetros y generando una batería de pruebas para obtener el modelo óptimo. Dado el bajo volumen de datos, se ha tratado de crear datos espurios a partir de los reales, con una distribución similar. Esto se ha hecho con el fin de que el modelo, que ya es complejo de por sí, disponga de una mayor cantidad de datos para poder entrenarse.

Los resultados obtenidos, a pesar de no ser todo lo precisos que un puediera desear, pueden servir para entender el impacto de la huella del hombre en la selva amazónica. La predicción establece que en 2030 se alcanzará una deforestación acumulada superior a 1M km^2.

Palabras clave

Amazonia, deforestación, preprocesamiento, red neuronal densa, red neuronal recurrente LSTM

Abstract

This TFG aims to make use of technology to carry out an analysis of the human impact on the environment. Firstly, it is intended to emphasize the importance of the Amazon rainforest on the planet and the excessive deforestation to which it is being subjected, which can be analyzed from the data and predicted by means of artificial intelligence. Next, a detailed explanation of the functioning of neural networks, fundamental in the designed model, is offered. Subsequently, the problem posed is detailed, which consists in predicting deforestation in the Amazon in the coming years. The following are exposed: data preprocessing, the designed model, implementation details and, finally, the results, conclusions and future work are exposed.

Specifically, the model tries to predict, through neural networks, the deforestation rate in the Amazon rainforest in the coming years. It is available a dataset consisting of 760 municipalities, including fixed data (latitude, longitude, area, hydrography, non-forest area, etc.) and annual deforestation data for the last 20 years. This is why the designed model combines dense neural networks for fixed variables and recurrent LSTM neural networks for temporal variables.

It is a complex problem, since it is a regressor, which generally involves more difficulty than a classifier. The variables available are fixed and temporary, which adds a further complication. In addition, the volume of the dataset is not desirable.

Many iterations have been performed testing different model configurations, adjusting hyperparameters, and generating a battery of tests to obtain the optimal model. Given the low volume of data, it has been attempted to create spurious data from the real data, with a similar distribution. This has been done so that the model, which is already complex in itself, has more data to train.

The results obtained, despite not being as precise as one might wish, can serve to understand the impact of man's footprint in the Amazon jungle.

Keywords

Amazon, deforestation, preprocessing, dense neural network, LSTM recurrent neural network

Índice

Listas

Lista de códigos

Lista de ecuaciones

Lista de figuras

1

Introducción

En esta sección introductoria trato de ofrecer una visión general de la motivación de este Trabajo de Fin de Grado y la estructura del mismo. En primer lugar, explico en detalle la importancia de la Amazonia para el planeta, ofreciendo datos concretos que justifican dicha importancia. Seguidamente expongo los objetivos de este Trabajo de Fin de Grado y, finalmente, detallo la estructura del documento.

1.1. La Amazonia

La Amazonia es el bosque tropical más extenso del mundo, situado en Sudamérica. Su superficie original oscila entre 5,5M y 6,7M km^2 según varias fuentes. La región amazónica pertenece a nueve países, de los cuales Brasil abarca la mayor extensión, seguido por Perú, Bolivia, Colombia, Venezuela, Ecuador, Guyana, Francia (Guayana Francesa) y Surinam. La selva amazónica destaca por la gran biodiversidad, ofreciendo una fauna y flora muy ricas en cantidad y variedad. Es el hogar del 10 % de todos los animales y plantas conocidas en la Tierra. Hay aproximadamente 40.000 especies de plantas, 400 mamíferos, 1.300 variedades de aves y una población de insectos que se cuenta por millones. [1]

“El Amazonas es el pulmón del planeta”. Todos hemos escuchado esta frase en algún momento a lo largo de nuestra vida. De hecho se estima que produce en torno a un 10-12 % del oxígeno del planeta. Sobra decir que se trata de un excepcional ecosistema que tenemos que cuidar para que las generaciones venideras puedan disfrutar de los muchos beneficios que ofrece, así como nosotros lo hacemos con esta maravilla que nos ha sido regalada.

Lamentablemente, en la actualidad no se están poniendo los medios necesarios para mantenerlo, explotándolo de un modo sostenible. Incendios, provocados o no, unidos a la tala masiva de árboles para la industria maderera y la creación de superficies para la agricultura, están acabando con la Amazonia. En los últimos años se está elevando la deforestación y esto puede llevarnos a una situación irreversible. Además, dada la coyuntura socioeconómica y política actual, las previsiones son pesimistas.

1.2. Objetivos

En estos últimos años se está produciendo una sensibilización generalizada de que tenemos un deber como habitantes del Planeta Tierra: cuidarlo para que las generaciones venideras puedan distrutar de él como lo hacemos nosotros y como lo hicieron nuestros antepasados previamente. Esto conlleva una actitud concreta, una forma de vivir en este mundo respetuosa con el medio ambiente. Cada uno puede incluir hábitos en su rutina para aportar su granito de arena: desde separar la basura para reciclaje hasta colaborar como voluntario recogiendo colillas en las playas, pasando por utilizar más el transporte público. Cierto es que todavía queda un largo camino por recorrer, pero sí es visible la voluntad de una gran mayoría social, sobre todo en países desarrollados, de que se tomen medidas de calado para vivir en este mundo de un modo más sostenible. Este sentir se está trasladando a los máximos dirigentes políticos a nivel mundial, quienes tienen la mayor capacidad para promulgar las leyes pertinentes.

Por ello, mi objetivo principal en este Trabajo de Fin de Grado es concienciar sobre la trascendencia del mantenimiento de la Amazonia y el futuro que le espera si seguimos actuando como lo hemos hecho en los últimos años. Tras lo explicado en la sección anterior, quizás uno puede ser más consciente de la importancia de la Amazonia para el planeta. Con la exposición de todo el trabajo realizado a nivel técnico para estimar la deforestación en los próximos años, trato de aportar datos concretos para justificar porvenir de la selva amazónica si no se revierte la situación.

1.3. Estructura del trabajo

Este Trabajo de Fin de Grado se divide en cuatro capítulos, los apartados de Bibliografía y Glosario y un anexo. A continuación, se procede a describir el contenido de cada uno de ellos:

En el primer capítulo se explica en detalle qué es la Amazonia, incluyendo datos concretos que justifican su relevancia en el planeta. Además, se expone cuál es la motivación principal que me ha inspirado a hacer este trabajo.

En el segundo, se aborda el estado del arte, ofreciendo una exposición de qué son las redes neuronales, cuáles sus fundamentos y cómo funcionan. También se incluye una descripción de las redes LSTM, un tipo de red neuronal recurrente que ha sido utilizada para trabajar con datos que varían a lo largo del tiempo.

El tercero es el más extenso, ya que contiene todo el trabajo realizado. Se divide en cuatro subsecciones que detallan la obtención y preprocesamiento de los datos, el diseño de la red neuronal, su desarrollo y las pruebas realizadas para configurar la red que minimiza el error.

El cuarto y último capítulo muestra los resultados de deforestación obtenidos por la red neuronal,

a partir de los cuales se extraen conclusiones para concienciar al lector del porvenir de la Amazonia. Además, se describe un posible trabajo futuro que puede continuar a partir del ya realizado.

Seguidamente se presenta un apartado con la bibliografía, que recoge las referencias utilizadas durante todo el proceso de investigación y configuración de la red neuronal. También se incluye un glosario con definiciones aclaratorias de términos que pueden resultar confusos al lector.

Por último, se puede encontrar un anexo que muestra un descriptivo de los datos sobre los que se ha trabajado.

2

Estado del arte: Redes neuronales

Las redes neuronales suponen el núcleo del Deep Learning. Son versátiles, potentes y escalables. Esto las hace ideales para abordar problemas de Machine Learning complejos, como la clasificación de imágenes (ej. Google Images), reconocimiento de voz (ej. Siri de Apple), recomendación de canciones (ej. Spotify) o aprender a derrotar al campeón mundial del juego Go tras jugar millones de partidas contra sí misma (AlphaZero de DeepMind). En mi caso, las utilizo para predecir la deforestación de la Amazonia en los próximos años.

Esta sección recoge un breve repaso histórico de las redes neuronales, seguido de una detallada explicación técnica, incluyendo cuáles son sus fundamentos y cómo funcionan. Además, se hace incapié en las redes LSTM, un tipo de red neuronal recurrente que ha sido de vital importancia para el desarrollo de este Trabajo de Fin de Grado.

2.1. Historia

Las redes neuronales artificiales fueron introducidas por primera vez en 1943 por el neurofisiólogo Warren McCulloch y el matemático Walter Pitts. Presentaron un modelo computacional simplificado sobre cómo las neuronas biológicas pueden trabajar de manera conjunta en cerebros animales para realizar cálculos complejos utilizando lógica proposicional. Esta fue la primera red neuronal artificial.

El éxito inicial hasta los años 60 llevó a pensar que pronto se lograría que los humanos pudiéramos conversar con máquinas inteligentes. Al no conseguirse pasado un tiempo, las redes neuronales perdieron protagonismo hasta principios de los 80, cuando se inventaron nuevas arquitectura y métodos de aprendizaje. El progreso no fue rápido y en los años 90 aparecieron otros potentes algoritmos de Machine Learning, como Support Vector Machine. Estos algoritmos ofrecían mejores resultados y fundamentos teóricos más sólidos, por lo que nuevamente las redes neuronales quedaron relegadas a un segundo plano.

Últimamente se ha recuperado el interés y parece que esta vez sí estamos en una buena situación para sacarle el máximo potencial a las redes neuronales, gracias a la gran cantidad de datos disponibles, el incremento de la capadidad computacional y los nuevos métodos de aprendizaje. Además se

ha financiado su investigación y, tras observar los buenos resultados obtenidos por algunos productos, se anima a más gente a seguir invirtiendo y trabajando con ello.

2.2. Fundamentos

Los pájaros nos inspiraron a volar, las bardanas inspiraron el velcro, e incontables invenciones más han sido inspiradas por la naturaleza. Parece lógico, pues, examinar la arquitectura del cerebro para inspirarnos a la hora de construir una máquina inteligente [2]. Es por esto que las redes neuronales artificales están basadas en las redes neuronales biológicas.

Las neuronas biológicas son unas células específicas que se encuentran principalmente en el córtex cerebral de los animales. Se componen del cuerpo celular, que contiene el núcleo y otros componentes celulares, y una o varias extensiones que transmiten impulsos hacia el cuerpo celular, llamadas dendritas. También cuentan con el axón, una prolongación más larga que conduce los impulsos desde el cuerpo celular hacia otras neuronas. En el extremo del axón, este se divide en varias ramas, cuyos extremos son las terminaciones sinápticas, que se conectan con las dendritas de otras neuronas. La principal diferencia entre neuronas y otras celulas es su longitud (gracias a las dendritas y los axones): las conexiones entre neuronas son posibles a larga distancia, mientras las demás celulas solo pueden conectar con vecinos cercanos. Por esto, la memoria puede ser asociativa o “por contenido”, en lugar de “por dirección”, y por ende, robustas respeto a ruidos, perdidas, errores, etc.

Estas neuronas, que por separado parecen cumplir una función simple, se organizan en una red de billones de neuronas, donde cada una se conecta con miles de neuronas. Estas se activan, transmitiendo información a las neuronas a las que están conectadas, en función de la tarea concreta que se desee realizar (ej. las neuronas se activan de una manera distinta para mover el brazo o para hablar). El trabajo conjunto es la clave de que podamos resolver problemas complejos. A pesar de que todavía se está investigando este campo, algunas partes del cerebro han sido mapeadas y parece que las neuronas se suelen agrupar en capas.

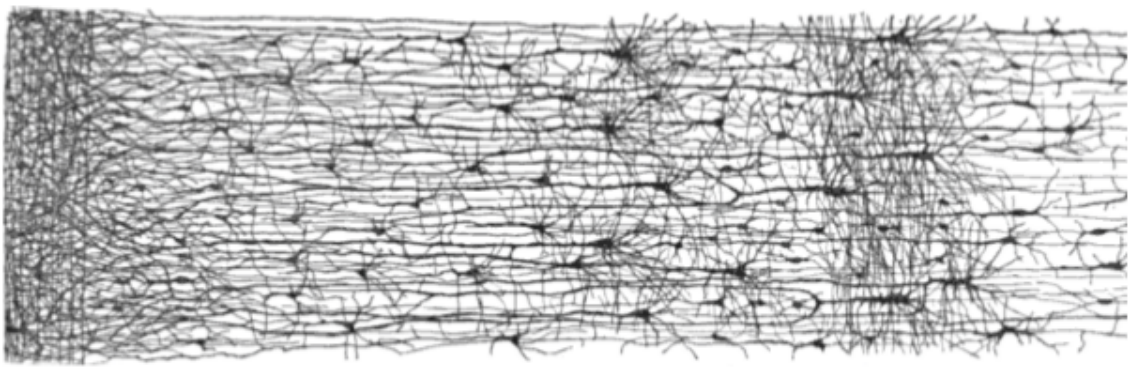


Figura 2.1: Múltiples capas en una red neuronal biológica [3]

Una vez entendido el fundamento biológico, y asumiendo que se ha producido un gran avance en la implementación las redes neuronales artificiales, veamos cómo funcionan en la actualidad.

2.3. Funcionamiento

Una red neuronal se puede entender como una función que, dado un input numérico n-dimensional, devuelve un output numérico m-dimensional. Dicha función es una composición de las funciones propias de cada neurona. Esta serie de cálculos desde la entrada hasta la salida se denomina Forward Pass.

Durante la fase de entrenamiento, se ajustan los parámetros de dicha función para minimizar el error. Esto se realiza mediante un algoritmo denominado Backpropagation.

2.3.1. Forward Pass

Supongamos la red neuronal más simple: el perceptrón. Este consta de una serie de variables input y una única neurona. Esta neurona recibe las distintas variables y el bias multiplicados por sus respectivos pesos. El bias funciona como una variable input más, con la particularidad de que su valor siempre es 1. Está conectado a cada neurona con un peso específico y sirve para controlar qué tan predispuesta está la neurona a activarse o no independiente de los pesos.

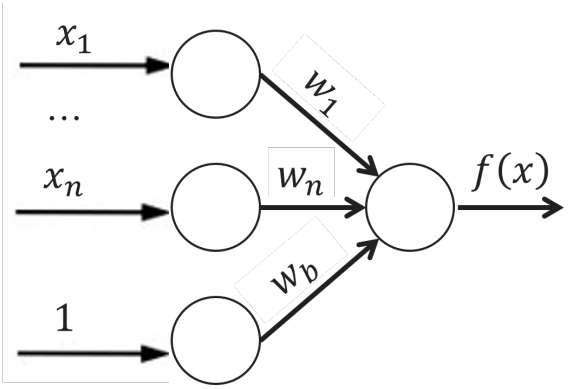


Figura 2.2: Representación gráfica de una neurona en un perceptrón

En la neurona se calcula el sumatorio de todos estos términos y se le aplica la función de activación:

$$f(x_1, ..., x_n) = \phi(\sum w_i x_i + w_b) \tag{2.1}$$

donde x_i es el valor de un input, w_i es el peso que multiplica a x_i y ϕ es la función de activación.

Otra forma de expresarlo con vectores es:

$$f(X) = \phi(X^t W + w_b) \tag{2.2}$$

donde $X = (x_1, ..., x_n)$ y $W = (w_1, ..., w_n)$ y ϕ es la función de activación.

Incluso, gracias al álgebra lineal, se pueden realizar los cálculos para hallar los outputs de una capa a partir de una serie de instancias simultáneamente de una manera eficiente:

$$f(X) = \phi(XW + b) \tag{2.3}$$

- X representa la matriz de inputs, con una fila por cada instancia y una columna por cada input.
- W es la matriz de pesos, que contiene todos los pesos excepto los de las neuronas bias. Tiene una fila por cada neurona input y una columna por cada neurona artificial.
- b es el vector bias, que contiene todos los pesos del entre la neurona bias y las neuronas artificiales de la capa.
- ϕ es la función de activación.

La función de activación se aplica al resultado del sumatorio para simular la activación de las neuronas biológicas, acotando el conjunto de valores de salida en un rango determinado, típicamente (0,1) ó (-1,1). Se buscan funciones cuyas derivadas sean simples, para minimizar con ello el coste computacional durante el Backpropagation. Este es un algoritmo fundamental para el aprendizaje de la red neuronal y que será explicado en detalle más adelante. Existen muchas funciones de activación, como la relu, sigmoidal o softmax.

Una vez entendido el funcionamiento de una neurona, el de la red neuronal no es más que la concatenación de neuronas, donde el output de una neurona se transmite a todas aquellas con las que está conectada, multiplicado por sus respectivos pesos. Cada neurona destino, tras recibir sus pertinentes inputs, hace los cálculos análogos y los transmite a aquellas neuronas con las que esté conectada, nuevamente multiplicados por sus respectivos pesos. Este proceso continúa hasta la neurona o neuronas output, donde se obtiene el resultado de la red neuronal.

Así, tras todo este proceso, se llega a una función compuesta por las funciones de cada neurona que simula la red neuronal, que consta de las variables input, los pesos y las funciones de activación. No parece muy complejo, pero ahora pueden surgir dudas sobre qué valor tienen los pesos y qué son exactamente las funciones de activación.

El ajuste de los pesos de una red neuronal se produce en el entrenamiento, mediante el algoritmo Backpropagation, que consiste en minimizar el error del output utilizando descenso por gradiente. Aquí las funciones de activación juegan un papel fundamental, ya que sus derivadas impactan de manera notable al gradiente. Veamos en detalle cómo funciona este algoritmo.

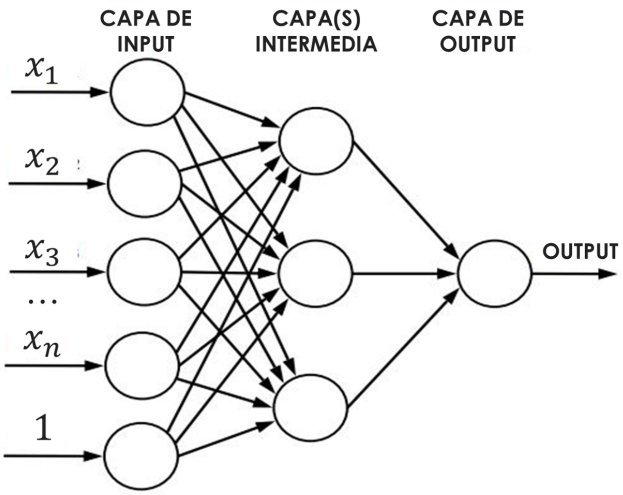


Figura 2.3: Representación gráfica de un perceptrón

2.3.2. Backpropagation

Supongamos que los pesos se inicializan con valores aleatorios. Al comenzar el entrenamiento, los primeros resultados obtenidos será previsiblemente erróneos, ya que son simplemente aleatorios. Para saber cuánto se ha equivocado la red neuronal, podemos comparar el output de la red neuronal con los valores esperados.

La función para calcular el error depende de los pesos y bias. Por ejemplo, en el perceptrón, que consta de una serie de neuronas input conectadas a una neurona output, dada una instancia de entrenamiento con inputs X y output y, la función de error o coste es:

$$C_{X,y}(W,b) = (f(X) - y)^2 = (\phi(X^t W + w_b) - y)^2 \tag{2.4}$$

Queremos minimizar el error medio de todo el conjunto de entrenamiento, de manera que se reduzca el error al calcular el output dado cualquier input. Para ello hallamos $C(W,b) = \frac{\sum_{i=1}^{n} C_{X_i,y_i}(W,b)}{n}$, siendo $n \in \mathbb{N}$ el número de elemetnos de entrenamiento, y buscamos $\nabla C(W,b) = 0$ para encontrar el mínimo. Para poder visualizarlo, supongamos que $C(w)$ solo depende de un solo peso w, luego buscamos $\frac{dC}{dw}(w) = 0$.

Funciones más complejas que la de la Figura 2.4 pueden tener varios extremos, ya sean máximos o mínimos, lo que puede dificultar la identificación del mínimo global. En la práctica, se ha demostrado que la función de error tiene la mayoría de mínimos acotados en una franja cuyo límite inferior es el mínimo global. Esto quiere decir que generalmente se alcanza un mínimo local de la misma calidad que el mínimo global. Además, aunque sea difícil encontrar el mínimo global, es irrelevante, ya que

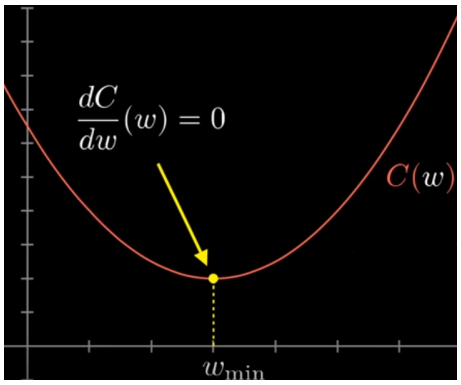


Figura 2.4: Mínimo de una función [4]

habitualmente da pie a overfitting. Por ello, nos centramos en encontrar un mínimo local. [5]

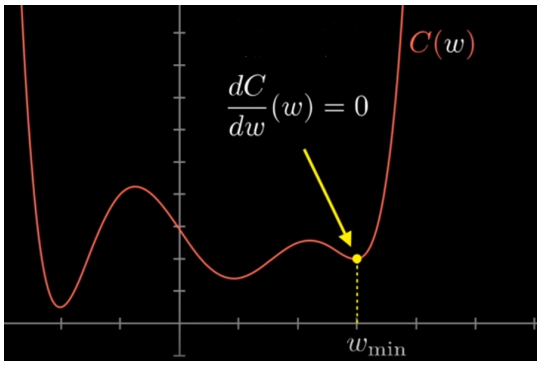


Figura 2.5: Mínimos de una función [4]

A medida que aumenta el número de parámetros de una función, la complejidad para encontrar un mínimo local es mayor. Esto es habitual en redes neuronales, que pueden depender de cientos de miles de pesos. Por ello, se plantea una táctica más flexible: el descenso por gradiente, que consiste en empezar en el punto obtenido al calcular el primer error medio e identificar hacia qué dirección debemos movernos, i.e. cómo cambiar los pesos, para acercarnos al mínimo. Para este ejemplo ilustrativo, habrá 2 posibilidades:

- Si la pendiente de la recta tangente es positiva, hacia la izquierda
- Si la pendiente de la recta tangente es negativa, hacia la izquierda

Repitiendo este proceso nos acabaremos acercando a un mínimo local. Es importante que los movimientos sean proporcionales a la pendiente de la recta tangente para acercarnos cada vez más al mínimo y evitar aumentar el error.

Ahora pensemos en 2 dimensiones. Entonces vamos a trabajar con $\nabla C(w_1, w_2)$. Sabemos que el vector gradiente tiene la propiedad de que en cada punto indica la dirección en la cual la función aumenta más rápidamente de valor y la dirección contraria indica la dirección de máxima pérdida [6]. Entonces $-\nabla C(w_1, w_2)$ representa hacia dónde movernos para alcanzar el mínimo más rápidamente. Se puede visualizar como una bola descendiendo por una ladera hacia el mínimo, tal y como se ilustra

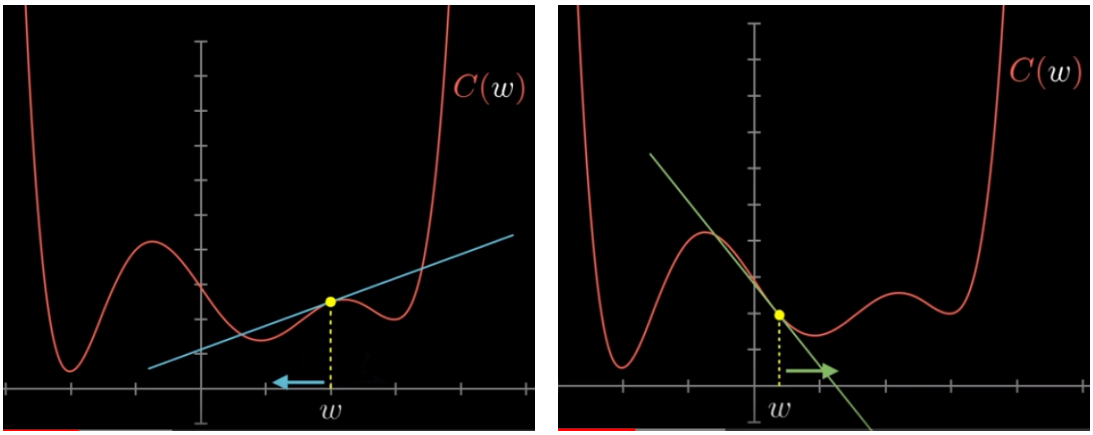


Figura 2.6: Descenso por gradiente [4]

a continuación:

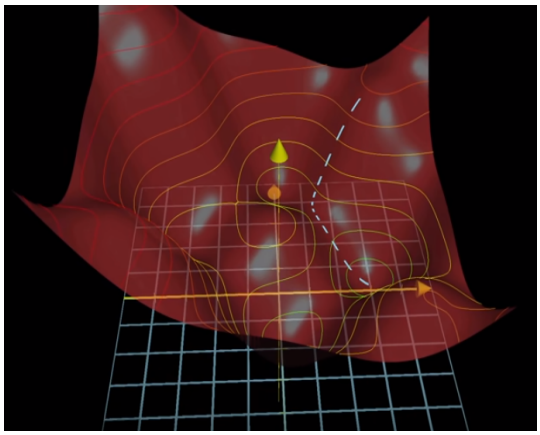


Figura 2.7: Descenso por gradiente [4]

Podemos elegir esta estrategia para cualquier función de dimensión $n \in \mathbb{N}$. Así, $-\nabla C(W,b) = -\nabla C(w_1,...,w_{n-r},b_1,...,b_r)$ indica cómo actualizar los pesos para mejorar la precisión de la red neuronal. Veamos cómo funciona Backpropagation en una red simple de L capas con una capa de entrada, $L-2$ capas intermedias y una de salida, con una neurona en cada capa:

En la neurona de salida, la función de error depende del peso que la conecta con la neurona anterior $w^{(L)}$, el bias $b^{(L)}$ y el input (output de la neurona anterior) $a^{(L-1)}$, que a su vez depende de sus respectivos peso, bias e input; y así sucesivamente hasta la primera capa. Dada una instancia de entrenamiento cualquiera con input x y output y, el error que compara la salida de la red con el valor esperado se calcula así:

$$\boxed{C_0 = C(w^{(1)},b^{(1)},...,w^{(L)},b^{(L)}) = (a^{(L)}-y)^2} \tag{2.5}$$

$$a^{(L)} = \phi(z^{(L)})$$

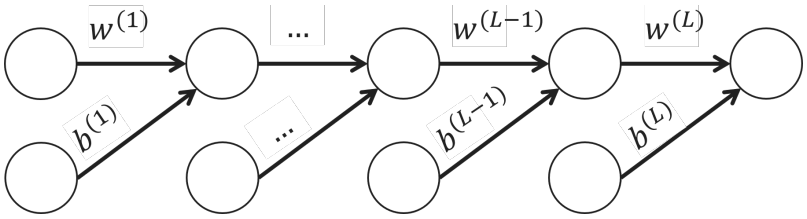


Figura 2.8: Red neuronal con una neurona por capa

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

Entonces, por la Regla de la Cadena, podemos hallar la derivada del error respecto del peso:

$$\boxed{\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-1)} \phi'(z^{(L)}) 2(a^{(L)} - y)} \tag{2.6}$$

$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \phi'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

De manera análoga, podemos calcular la derivada del error respecto del bias:

$$\boxed{\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = \phi'(z^{(L)}) 2(a^{(L)} - y)} \tag{2.7}$$

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1$$

Así, podemos iterar para todas las capas y hallar los ajustes a realizar para todos los pesos y bias. Por ejemplo, para hallar el ajuste de $w^{(L-1)}$, necesitamos "propagar hacia atrás" aplicando la Regla de la Cadena:

$$\frac{\partial C_0}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} = a^{(L-2)} \phi'(z^{(L-1)}) w^{(L)} \phi'(z^{(L)}) 2(a^{(L)} - y) \tag{2.8}$$

Ahora veamos que generalizando para una red neuronal con $n_l \in \mathbb{N}$ neuronas en cada capa l, la idea es la misma. Para ello utilizamos la siguiente notación:

- $w_{jk}^{(l)}$: peso que conecta la neurona j de la capa l con la neurona k de la capa $l-1$
- $b_j^{(l)}$: bias de la neurona j de la capa l

$$\boxed{C_0 = \sum_{j=0}^{n_L - 1} (a_j^{(L)} - y_j)^2} \tag{2.9}$$

$$a_j^{(L)} = \phi(z_j^{(L)})$$

$$z_j^{(L)} = w_{j0}^{(L)} a_0^{(L-1)} + w_{j1}^{(L)} a_1^{(L-1)} + ... + w_{jn_{L-1}}^{(L)} a_0^{(L-1)} + b_j^{(L)}$$

Derivando respecto de los pesos de la última capa, aplicando la Regla de la Cadena, obtenemos:

$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} = a_j^{(L-1)} \phi'(z^{(L)}) 2(a_j^{(L)} - y_j) \tag{2.10}$$

$$\frac{\partial C_0}{\partial a_j^{(L)}} = 2(a_j^{(L)} - y_j)$$

$$\frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} = \phi'(z_j^{(L)})$$

$$\frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}} = a_j^{(L-1)}$$

Como vemos es muy similar a la red simple con una neurona en cada capa, pero al "propagar hacia atrás" el error, hay que tener en cuenta que el output de la neurona k de una capa $l < L$ se transmite a varias neuronas de la capa $l + 1$. Por ejemplo:

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L - 1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}} = \sum_{j=0}^{n_L - 1} w_{jk}^{(L)} \phi'(z_j^{(L)}) \frac{\partial C_0}{\partial a_j^{(L)}} \tag{2.11}$$

Con todo, finalmente llegamos a calcular el gradiente con todas sus derivadas parciales:

$$\frac{\partial C_0}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \phi'(z_j^{(l)}) \frac{\partial C_0}{\partial a_j^{(l)}} \tag{2.12}$$

Donde $\frac{\partial C_0}{\partial a_j^{(l)}}$ se calcula de una forma distinta si l es la última capa o no:

$$\frac{\partial C_0}{\partial a_j^{(l)}} = \begin{cases} \sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \phi'(z_j^{(l+1)}) \frac{\partial C_0}{\partial a_j^{(l+1)}} & si \quad l < L \\ 2(a_j^{(L)} - y_j) & si \quad l = L \end{cases}$$

Observamos que para que existan las derivadas parciales, la función de activación ϕ ha de ser derivable. Para que funcione el Descenso por Gradiente es fundamental elegir una ϕ tal que $\phi' \neq 0$ y sea fácil de calcular para minimizar el coste computacional. Aun así, también se suele trabajar con algunas que no son derivables en el 0, como la relu, definiendo la derivada en el 0 artificialmente. Esto se debe principalmente a que en la práctica es capaz de obtener buenos resultados con una

buena eficiencia computacional. En la Figura 2.9 podemos observar algunos ejemplos de funciones de activación con sus respectivas derivadas: tangente hiperbólica y sigmoidal/logística (derivables en 0); relu y escalón (no derivables en 0).

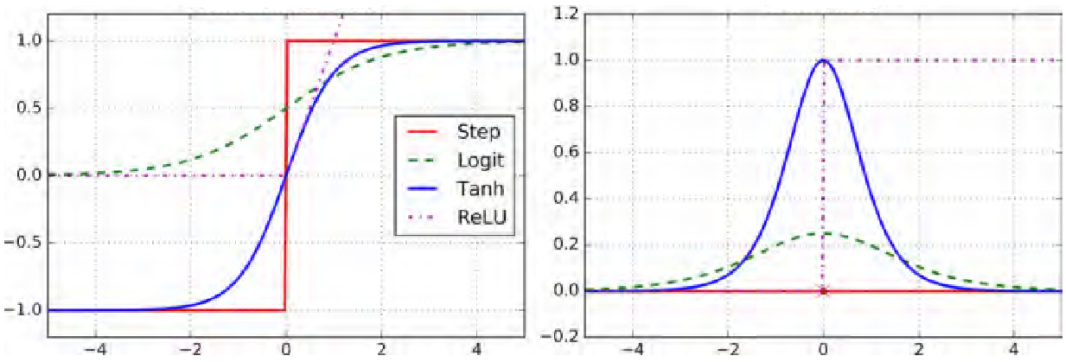


Figura 2.9: Funciones de activación [2]

Hemos realizado estos cálculos para una instancia de entrenamiento, pero nos interesa calcular la media de las derivadas del error para todo el conjunto de entrenamiento. De este modo, actualizaremos los pesos y bias minimizando el error medio:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial w^{(L)}} \quad , \quad \frac{\partial C}{\partial b^{(L)}} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial b^{(L)}} \tag{2.13}$$

Queremos que la red neuronal aprenda de todo el conjunto de entrenamiento, pero, por razones computacionales, generalmente no se actualizan los pesos tras evaluar todas las instancias. Por contra, se reordena todo el conjunto de entrenamiento de manera aleatoria y se divide en batches de un tamaño determinado. La actualización de pesos se produce aplicando el descenso por gradiente medio de los elementos del batch. Así, no estamos actualizando los pesos de manera que minimicemos el error medio de todas las instancias del conjunto de entrenamiento, sino de las que se encuentran en el batch. Esto da pie a que no se calcule el mejor descenso posible, pero sí una aproximación que es mucho más eficiente desde el punto de vista computacional.

Al principio hemos comentado que los pesos se inicializan de manera aleatoria, pero cabe la pena mencionar que también pueden inicializarse con valores prefijados, con el fin de que la red converja más rápidamente al mínimo error. Esto se suele hacer por medio de una técnica llamada Transfer Learning, que consiste en inicializar los pesos con los de un modelo preentrenado que se ha utilizado para un problema similar. Por ejemplo, si hemos entrenado una red neuronal para clasificar imágenes de animales y queremos desarrollar un clasificador de imágenes de coches, podemos utilizar Transfer Learning, ya que ambas tareas son similares.

Finalmente, resumamos lo visto en esta sección: para cada instancia de entrenamiento el algoritmo Backpropagation primero hace una predicción (forward pass) y evalúa el error comparándolo con el

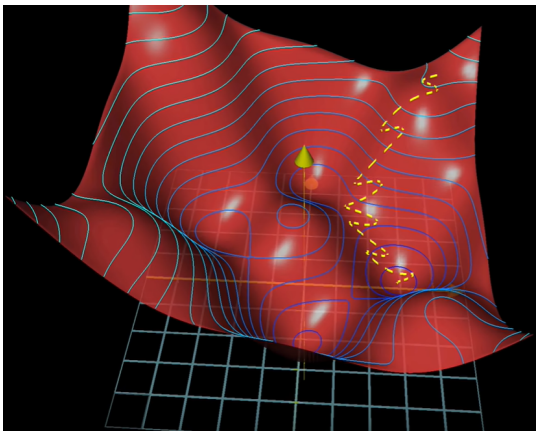


Figura 2.10: Descenso por gradiente calculado con batches [7]

valor real, después se propaga hacia atrás a través de las distintas capas caclulando la contribución de cada conexión al error, y finalmente ajusta los pesos y bias para reducir el error por medio del Descenso por Gradiente.

Hemos visto el funcionamiento de una red neuronal básica: el perceptrón. Hay muchos tipos de redes neuronales más complejas, que también se basan en este funcionamieno. En la siguiente sección vamos a entender las redes LSTM, que son muy útiles para abordar problemas de series temporales.

2.4. Redes neuronales recurrentes: LSTM

En este Trabajo de Fin de Grado se trabaja con un conjunto de datos de deforestación anual. Esto quiere decir que disponemos de una serie de datos que siguen un orden temporal.

Para abordar problemas de series temporales se suelen utilizar redes neuronales recurrentes, que tienen en cuenta los datos desde t_0 a t_{n-1} para predecir el valor en t_n. En realidad no son tan diferentes de una red neuronal normal. Una red neuronal recurrente puede considerarse como copias múltiples de la misma red, cada una de las cuales transmite información a su sucesora. Se trata de una red con un bucle, lo cual permite que la información persista. En la Figura 2.12 se puede observar lo que sucede si se desenrolla en bucle:

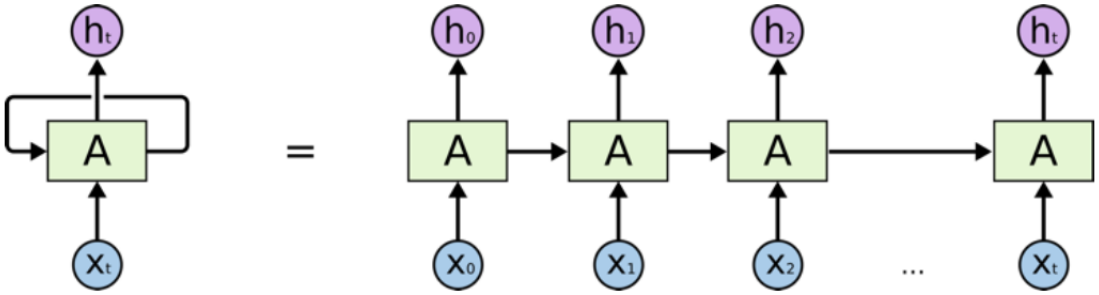


Figura 2.11: Red neuronal recurrente desenrollada [8]

Concretamente, la red neuronal recurrente A recibe la secuencia de inputs temporales X_i, siendo $i < n \in \mathbb{N}$. Primero recibe el input X_0 y genera el output h_0. En el siguiente paso, los inputs son h_0 y X_1 y se genera el output h_1, que nuevamente sirve de input en el siguiente paso junto con X_2. Este proceso se repite sucesivamente hasta el último paso, que recibe el input X_n y el output del paso anterior h_{n-1} para predecir h_n, que es el output del modelo. En definitiva, una red recurrente evalúa cada input X_i y output del paso anterior h_{i-1} para generar el output h_i y transmitirlo al siguiente paso.

Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetitivos de red neuronal. En las redes recurrentes estándar, este módulo repetitivo tiene una estructura muy simple, como por ejemplo una sola capa con la tangente hiperbólica como función de activación.

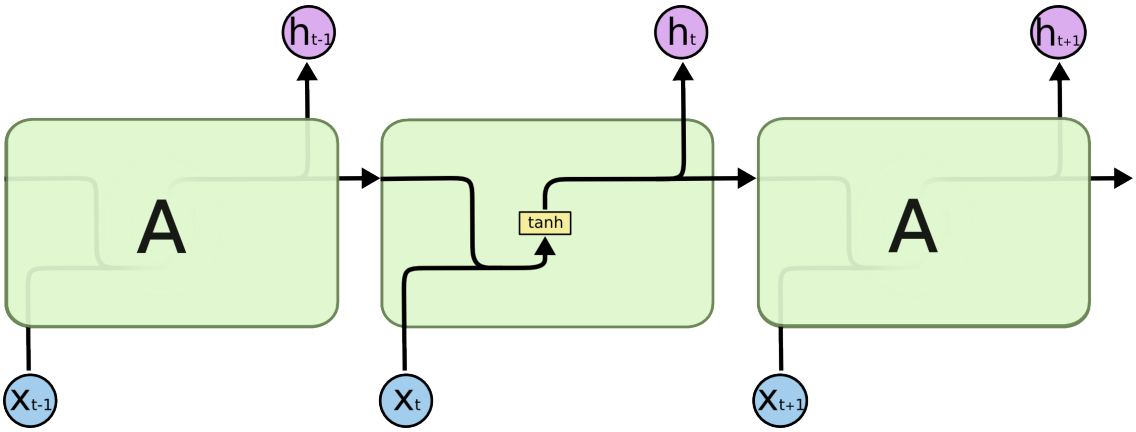


Figura 2.12: Red neuronal recurrente [8]

Un problema que presentan las redes neuronales recurrentes estándar es que en la práctica no funcionan bien con dependecias temporales largas, ya que el resultado final está influenciado en gran medida por los últimos pasos. Además, durante el entrenamiento, dado que la información se procesa a través de un bucle, las actualizaciones de los pesos son drásticas. La acumulación del gradiente del error en un bucle puede dar lugar a dos escenarios extremos:

- Explosión: los gradientes tienen valores mayores que 1, que al multiplicarlos sucesivamente resultan en un valor

excesivamente alto, que puede provocar overflow y que la red retorne valores Nan.

- Desvanecimiento: los gradientes tienen valores menores que 1, que al multiplicarlos sucesivamente resultan en un valor excesivamente bajo, de modo que los pesos de la red apenas se actualizan y la red no aprende.

Esto ha dado pie a que se hayan investigado nuevas configuraciones de redes neuronales recurrentes, donde recientemente destacan las LSTM (Long-Short Term Memory), que disponen de puertas de control de memorización para lidiar con estos problemas. Estas son capaces de tener en cuenta más información, pudiendo dar un peso mayor a los pasos anteriores en lugar de priorizar siempre los últimos. De este modo son capaces de hallar dependecias en el corto y largo plazo temporal, que supone la principal ventaja respecto a las redes recurrentes estándar.

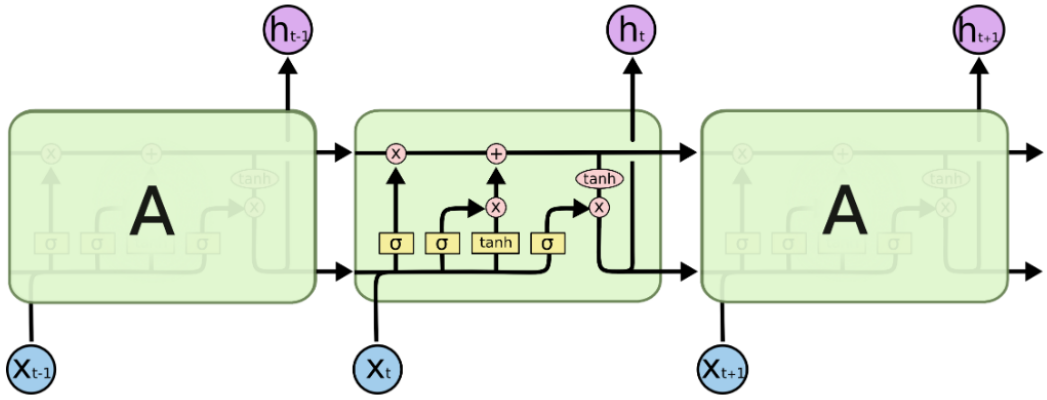


Figura 2.13: Red neuronal LSTM [8]

Inicialmente, X_t y h_{t-1} se concatenan y pasan por una capa sigmoidal llamada “forget gate”, que devuelve un vector de valores entre 0 y 1 de la misma longitud que el estado C_{t-1}. Esta capa sirve para filtrar qué se debe mantener del estado anterior. Un 1 significa que esa información se tiene en cuenta en su totalidad, mientras que un 0 quiere decir que se ignora por completo.

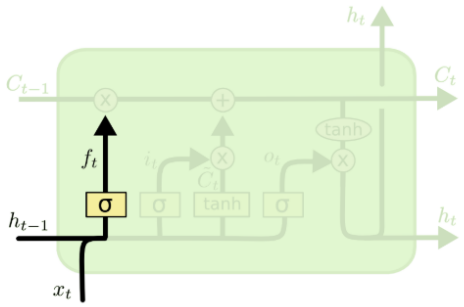


Figura 2.14: Forget gate [8]

$$f_t = \sigma(W_f[h_{t-1}, Xt] + b_f) \tag{2.14}$$

En la capa “input gate”, que es otra capa sigmoidal, se decide qué valores se van a actualizar en el estado. Además, por otro lado, en una capa tanh se crea un vector candidato $\tilde{C}_t$, servir para actualizar C_t. Posteriormente, se multiplican las salidas de estas dos capas para crear una actualización del estado.

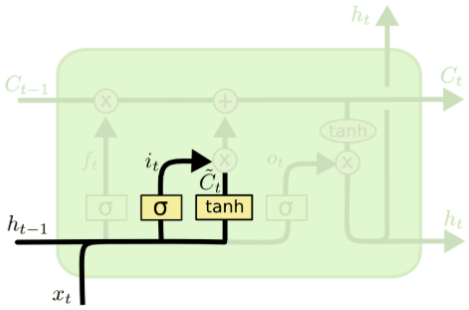


Figura 2.15: Input gate [8]

$$i_t = \sigma(W_i[h_{t-1}, Xt] + b_i) \quad , \quad \tilde{C}_t = tanh(W_C[h_{t-1}, Xt] + b_C) \tag{2.15}$$

A continuación se actualiza C_{t-1} al nuevo valor C_t. Para ello se multiplica el estado anterior por f_t, desechando la información irrelevante, y luego se le suma $i_t * \tilde{C}_t$ para actualizar el estado.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.16}$$

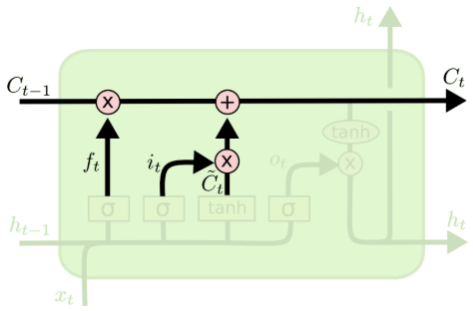


Figura 2.16: Actualización del estado [8]

Por último, se genera la salida h_t, que depende del estado actual. Para ello, la capa sigmoidal "output gate" decide qué valores del estado van a formar parte del output. Se aplica tanh a los valores de C_t para normalizarlos entre -1 y 1 y se multiplican por la salida del "output gate", de manera que h_t recoge la información que el "output gate" ha considerado.

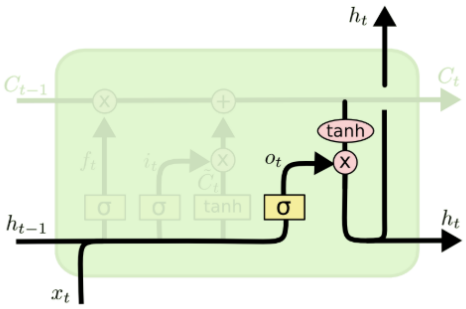


Figura 2.17: Output [8]

$$o_t = \sigma(W_o[h_{t-1}, Xt] + b_o)h_t = o_t * tanh(C_t) \tag{2.17}$$

En resumen, la clave de las redes LSTM es el estado, que se actualiza para saber qué tiene que recordar de los pasos anteriores, cómo se debe actualizar y cuáles son los valores relevantes para el output. En la práctica, estas mejoran significativamente los resultados de las redes neuronales recurrentes tradicionales. Esto justifica su uso para tratar de predecir los niveles de deforestación en la selva amazónica en los próximos años.

3

Trabajo realizado

Se ha desarrollado un modelo de predicción de deforestación amazónica con redes neuronales. Para ello, ha sido necesario encontrar un dataset que recoja el histórico de deforestación, preprocesar los datos para que sirvan como input al modelo y configurar una red neuronal sea capaz de interpretar los datos de manera adecuada para poder realizar las predicciones.

El conjunto de datos consta de 760 municipios situados en la selva amazónica, con una serie de variables fijas, como el estado al que pertenecen o el área que ocupan, y datos de deforestación anuales. Para que el modelo pueda interpretar estas 2 tipologías de datos, se ha diseñado un modelo que consta de una red neuronal recurrente LSTM para tratar los datos temporales combinada con una red densa para tratar datos fijos.

Dado que inicialmente solo se contaba con 760 municipios, se ha aplicado la técnica de *data augmentation* con el fin de crear datos falsos con una distribución similar a la de los datos reales, aumentando así el volumen del conjunto de datos de entrenamiento.

Este código ha sido íntegramente desarrollado en un *notebook* de Jupyter en *python*, haciendo uso de algunas librerías muy útiles, como *pandas* para trabajar con tablas de datos, *sklearn* para el preprocesamiento de datos o *keras* para la implementación de redes neuronales.

3.1. Obtención de datos y preprocesamiento

El conjunto de datos utilizado recoge el histórico de datos de deforestación de 760 municipios desde 2008 a 2019, obtenido del INPE, del portugués Instituto Nacional de Pesquisas Espaciais [9]. Conforme a las políticas definidas y actualizadas por el Ministerio de Ciencia y Tecnología brasileño, el INPE tiene como objetivo la promoción y ejecución de estudios e investigaciones científicas, el desarrollo tecnológico, la ejecución de actividades operacionales y la capacitación de recursos humanos en los campos de la ciencia espacial, observación de la Tierra, fenómenos atmosféricos, previsión meteorológica, estudios climáticos y otras áreas relacionadas.

3.1.1. Obtención de datos

En otoño de 2019 se encontraba disponible en la página del INPE un histórico de datos que recoge la deforestación entre 2000 y 2018, ya que los datos de 2019 no se encontraban disponibles. Estos constan de un archivo csv para cada año en el que se muestran 760 municipios con las siguientes variables:

- Variables fijas: no varían de año en año.
  - Municipio: nombre del municipio.
  - Estado: estado al que pertenece el municipio. Hay 9 estados, identificados por 2 letras:
    - Amazonas: “AM”
    - Roraima: “RR”
    - Acre: “AC”
    - Rondonia: “RO”
    - Mato Grosso: “MT”
    - Amapá: “AP”
    - Pará: “PA”
    - Tocantins: “TO”
    - Maranhão: “MA”
  - Latitud: latitud geográfica del municipio
  - Longitud: longitud geográfica del municipio Área total: área que pertenece al municipio
  - No bosque: área que no es bosque (montañas, tierra, etc)
  - Hidrografía: área cubierta por agua (ríos o lagos)
- Variables anuales: varían de año en año.
  - Deforestación: área de deforestación total acumulada hasta ese año
  - Incremento deforestación: área deforestación que se ha producido en ese año
  - Bosque: área de bosque en ese año
  - Nubes: área cubierta por nubes (no se sabe qué hay bajo las nubes) en ese año
  - No observado: área no observada ese año
  - Check: variable que compara la suma de todas las variables de área con el área total (útil para comprobar la consistencia de los datos)

A comienzos de 2020 a se actualizaron los datos, incluyendo los de 2019. En este periodo se han realizado cambios en la web y se han subido los datos en otro formato, recogiendo únicamente los incrementos de deforestación entre 2008 y 2019. Para los años entre 2008 y 2018, los datos varían ligeramente entre los datos obtenidos inicialmente y los nuevos datos actualizados. Se ha optado por sustituir los datos anteriores por los más actualizados, asumiendo que estos son más precisos. Se han añadido los datos para 2019 de la siguiente manera:

- Deforestación 2019 = Deforestación 2018 + Incremento deforestación 2019
- Bosque 2019 = Bosque 2019 - Incremento deforestación 2019
- Nubes 2019 = 0
- No observado 2019 = 0

- Check = 100

Combinando ambos conjuntos de datos, se ha logrado recoger un histórico de datos más extenso y lo más actualizado posible. Se trata de un conjunto de datos en formato *dataframe* de la librería *pandas*. Este cuenta con 760 filas, donde cada una recoge la información de cada municipio. Las primeras 7 columnas son las variables fijas y el resto son variables que registran cambios anualmente, donde se encuentran 6 variables para cada año entre 2000 y 2019. Tras la unificación de los datos, el *dataframe* a partir del cual se ha desarrollado el modelo tiene el siguiente aspecto:

	Municipio	Estado	Latitud	Longitud	Area total	No bosque	Hidrografia	Deforestacion 2000	Incremento deforestacion 2000	Bosque 2000	Nubes 2000	No observado 2000	Check 2000
0	NOVA COLINAS (MA)	MA	-7.17281	-46.20838	748.0	748.8	0.0	0.0	NaN	0.0	0.0	750.0	200.0
1	NOVO SANTO ANTÔNIO (MT)	MT	-12.32775	-50.97263	4375.0	3593.3	95.7	107.8	NaN	578.0	0.0	0.2	99.0
2	BOM JARDIM (MA)	MA	-3.96461	-46.50727	6648.0	55.9	7.5	2342.6	NaN	4241.8	0.0	0.2	100.0
3	ARAGUACEMA (TO)	TO	-8.90941	-49.57375	2786.0	2232.0	84.1	34.1	NaN	435.8	0.0	0.0	99.0
4	AXIXÁ (MA)	MA	-2.81750	-44.08819	205.0	0.0	57.2	12.1	NaN	135.7	0.0	0.0	100.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
755	CENTRO NOVO DO MARANHÃO (MA)	MA	-3.06829	-46.47701	8329.0	0.0	0.3	1758.7	NaN	6569.8	0.2	0.0	100.0
756	PARAGOMINAS (PA)	PA	-3.12356	-47.40354	19465.0	7.1	36.0	7212.3	NaN	12208.2	0.0	1.4	100.0
757	MOCAJUBA (PA)	PA	-2.54950	-49.35737	872.0	168.2	74.9	470.5	NaN	158.4	0.0	0.0	100.0
758	SÃO PAULO DE OLIVENÇA (AM)	AM	-4.73714	-69.52392	20467.0	86.3	587.8	103.8	NaN	19688.8	0.0	0.3	100.0
759	SENADOR LA ROCQUE (MA)	MA	-5.31274	-47.07804	1247.0	0.0	0.0	1092.0	NaN	155.0	0.0	0.0	100.0

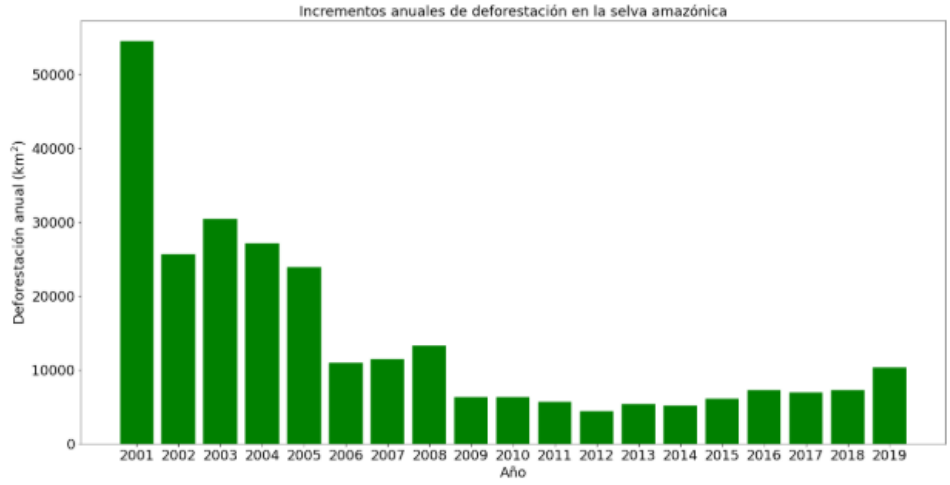
760 rows × 127 columns

Figura 3.1: Muestra parcial del *dataframe*

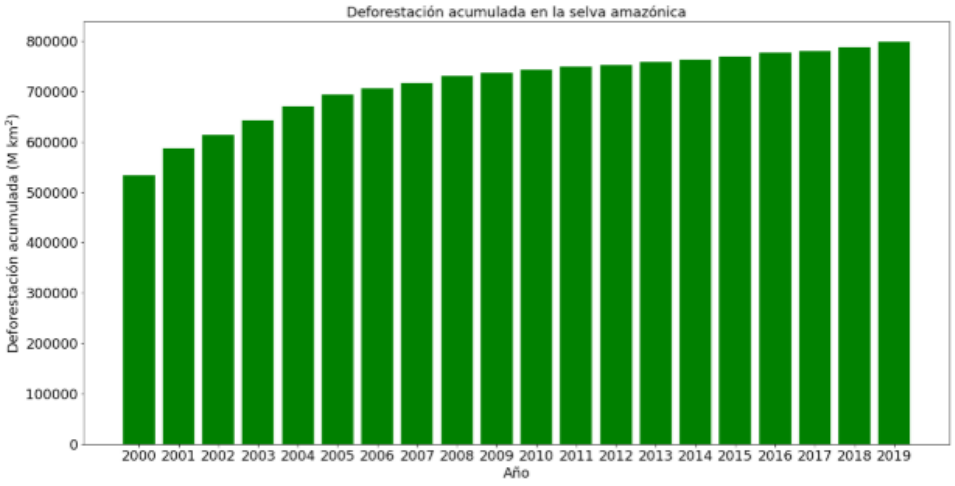
Este *dataframe* cuenta con 760 filas, una por municipio, y 127 columnas: 7 fijas + 6x20 variables entre 2000 y 2019. En el primer año, la variable “Incremento de deforestación” es *NaN*, ya que no se dispone de un año anterior con el que calcular el incremento. Este dato se omite en el modelo.

Agregando los datos de deforestación de los municipios, podemos entender la deforestación que se ha producido en la Amazonia entre 2000 y 2019. Se han calculado los incrementos de deforestación anuales y la deforestación total acumulada en cada año. Para ello se ha hecho uso de la función *sum* de *pandas*, En la Figura 3.2 se puede observar una importante disminución desde 2000 a 2012, pero a partir de ese mínimo se ha producido una tendencia alcista. Dada la actual coyuntura política, social y económica en Brasil, es difícil pensar que se van a poner los medios para frenar esta tendencia.

A partir de la latitud y longitud de cada uno de los 760 municipios, se han representado los municipios en un mapa geográfico, que se encuentra en la Figura 3.3. Para tal fin, se ha utilizado la librería *gmplot* de Google Maps [10].



(a) Incrementos anuales de deforestación en la selva amazónica



(b) Deforestación acumulada en la selva amazónica

Figura 3.2: Deforestación en la Amazonia entre 2000 y 2019

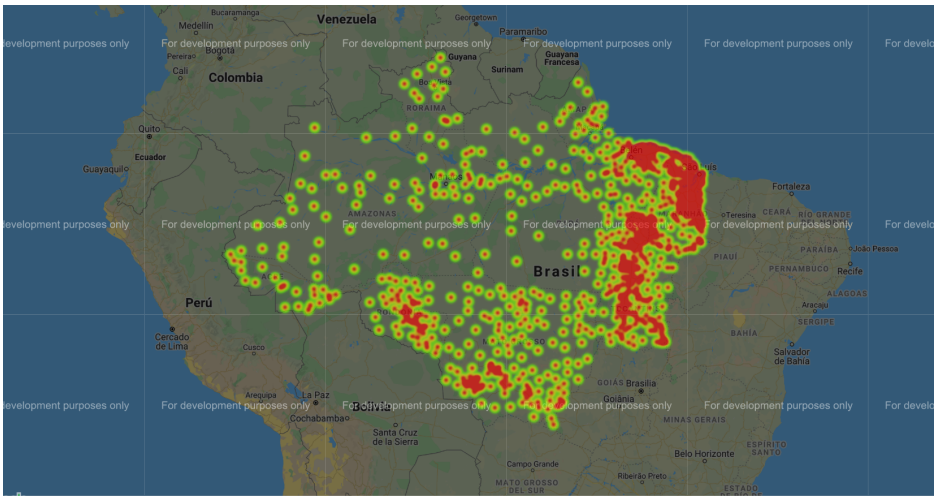


Figura 3.3: Mapa de calor de los municipios de la Amazonia

Se puede observar que en la zona sur y este de la Amazonia hay más densidad de municipios. Esto va en línea con el mapa de deforestación que proporciona el propio INPE, mostrado en la Figura 3.4. Así, se puede concluir que, donde más municipios hay, mayor es la deforestación.

Figura 3.4: Mapa de deforestación de la Amazonia [9]

El area total cubierta por los 760 municipios es: 5.068.048 km^2, mientras el area total de la Amazonia, según la fuente y el año, se estima entre 5,5 y 6,7 km^2. Por ello, podemos afirmar que estamos cubriendo una superficie que supone entre el 76 % y el 92 % del total. Dado este alto porcentaje, no es excesivamente osado extrapolar las conclusiones de este trabajo al total de la selva amazónica.

3.1.2. Preprocesamiento de datos

Según el algoritmo utilizado para desarrollar un modelo de Machine Learning, se requiere de un preprocesamiento de los datos adecuado con el fin de que el algoritmo de apredizaje funcione de manera correcta. En este caso, dado que se dispone de 2 tipologías de datos distintas, estas se han de estructurar de manera diferente para que sirvan de input a las distintas redes de las que se compone el modelo. Para que una red neuronal sea capaz de interpretar datos correctamente, estos deben ser numéricos, luego hay que transformar variables categóricas en numéricas. Además, los datos han de ser normalizados para evitar que unas variables destaquen por encima de otras, evitando así sesgos no deseados en el modelo.

En primer lugar creamos 2 subconjuntos de datos: el de variables fijas y el de deforestación anual. Ambos se pueden relacionar por el índice del *dataframe* o por el nombre del municipio. Cada uno es tratado de manera distinta:

- Datos fijos:
    1.– *One Hot Encoding* de la variable “Estado” por medio de la función *get_dummies* de *pandas*.
    2.– Normalización para acotar el resto de variables entre 0 y 1 utilizando la función *MinMaxScaler*

de *sklearn*.

3.– Almacenamiento en un array de *numpy* de tamaño 760x14: 760 municipios con sus respectivas variables “Latitud”, “Longitud”, “Área total”, “No bosque”, “Hidrografía” y 9 posibles estados (“PA”, “MA”, “TO”, “RO”, “AP”, “MT”, “AM”, “AC”, “RR”). Este servirá de input a la red densa.

	Municipio	Estado	Latitud	Longitud	Area total	No bosque	Hidrografia
0	NOVA COLINAS (MA)	MA	-7.17281	-46.20838	748.0	748.8	0.0
1	NOVO SANTO ANTÔNIO (MT)	MT	-12.32775	-50.97263	4375.0	3593.3	95.7
2	BOM JARDIM (MA)	MA	-3.96461	-46.50727	6648.0	55.9	7.5
3	ARAGUACEMA (TO)	TO	-8.90941	-49.57375	2786.0	2232.0	84.1
4	AXIXÁ (MA)	MA	-2.81750	-44.08819	205.0	0.0	57.2

Figura 3.5: Muestra parcial de los datos fijos del *dataframe* antes del preprocesamiento

- Datos temporales: En la red LSTM, que es la que recibe los datos anuales de deforestación, se debe definir el número de años. Por ello, se establece *window_size*, de manera que la red utitliza los datos de *window_size*-1 años para predecir la deforestación del años siguiente.

  1.– Normalización para acotar las variables entre 0 y 1 utilizando la función *MinMaxScaler* de *sklearn*

  2.– Almacenamiento en un array de *numpy* de tamaño 760x6x *window_size-1*: 760 municipios con sus respectivas variables “Deforestación”, “Incremento deforestación”, “Bosque”, “Nubes”, “No observado”, “Check” para *window_size*-1 años. Este servirá de input a la red LSTM

  3.– Almacenamiento en un array de *numpy* de tamaño 760: los valores de “Incremento deforestación”, que es la variable a predecir, en el año *window_size*. Este servirá para contrastar el output del modelo con los valores reales, con el fin de utilizar aprendizaje supervisado en el entrenamiento del modelo.

	Municipio	Deforestacion 2000	Incremento deforestacion 2000	Bosque 2000	Nubes 2000	No observado 2000	Check 2000	Deforestacion 2001	Incremento deforestacion 2001	Bosque 2001	Nubes 2001	No observado 2001	Check 2001
0	NOVA COLINAS (MA)	0.0	NaN	0.0	0.0	750.0	200.0	0.0	0.0	0.0	0.0	750.0	200.0
1	NOVO SANTO ANTÔNIO (MT)	107.8	NaN	578.0	0.0	0.2	99.0	122.8	15.0	563.0	0.0	0.2	100.0
2	BOM JARDIM (MA)	2342.6	NaN	4241.8	0.0	0.2	100.0	3079.1	736.5	3505.3	0.0	0.2	100.0
3	ARAGUACEMA (TO)	34.1	NaN	435.8	0.0	0.0	99.0	38.5	4.4	431.4	0.0	0.0	99.0
4	AXIXÁ (MA)	12.1	NaN	135.7	0.0	0.0	100.0	13.8	1.7	134.0	0.0	0.0	100.0

Figura 3.6: Muestra parcial de los datos variables del *dataframe* antes del preprocesamiento

La variable “Municipio” no se utiliza como input del modelo, pero sirve para identificar sus respectivos datos fijos y variables. Se almacena para luego identificar las predicciones de deforestación de cada municipio.

3.2. Diseño del modelo

El modelo diseñado se compone de 2 redes neuronales distintas que tratan por separado las variables fijas y las temporales. El output de ambas se concatena, se añade alguna capa densa adicional y se obtiene el output: la predicción de deforestación en el siguiente año. En la Figura 3.7 se puede observar una aproximación ilustrativa del modelo.

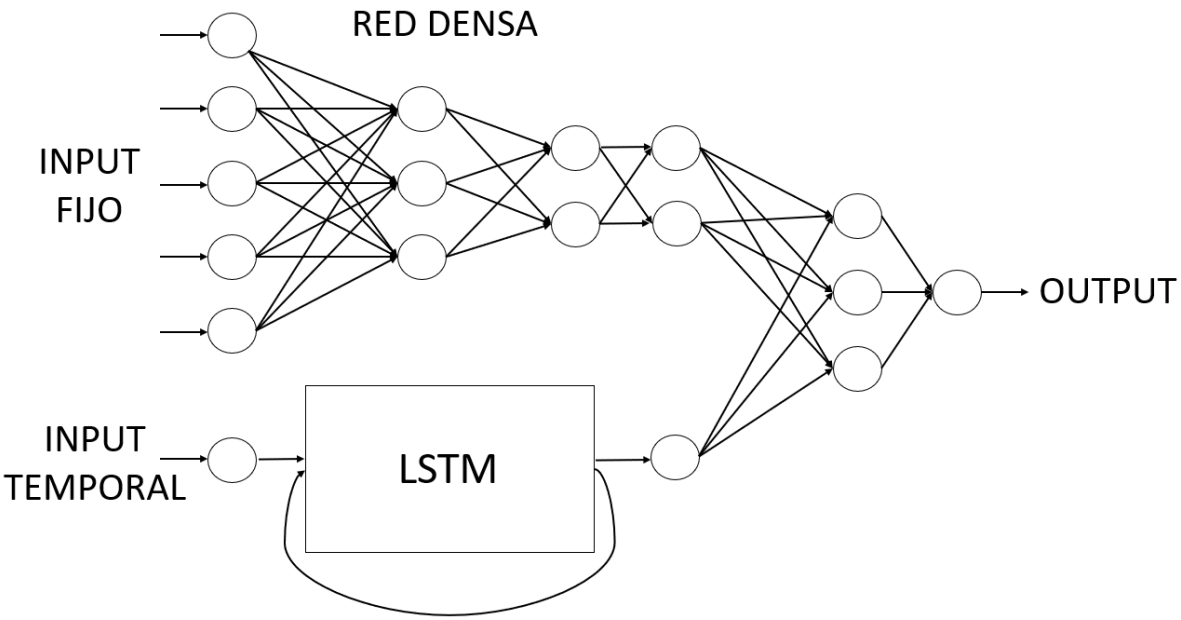


Figura 3.7: Diseño del modelo

1.– Red densa: recibe como input las variables fijas de un municipio, las que se mantienen constantes a o largo del tiempo.

2.– Red LSTM: recibe como input datos de deforestación de un municipio correspondientes a los *window_size*-1 años previos al año que se predice. Estos datos pasan por la red LSTM recursivamente *window_size*-1 veces.

3.– Concatenación: los outputs de la red densa y la red LSTM se concatenan en una capa, a la que se añade alguna capa adicional antes de ofrecer el output del modelo.

Dado que se dispone de datos fijos y variables, se ha decidido diseñar un modelo que combine redes neuronales capaces de trabajar con las 2 tipologías de datos. Se trata de un modelo complejo y original, lo que conlleva un mayor dificultad para obtener unos resultados base aceptables sobre los que iterar en busca de mejoras en el modelo.

Se ha realizado una batería de pruebas para escoger la mejor configuración posible: número de capas, tamaño de los batches, número de épocas, *window_size*, etc. Esto se detalla en la Sección 3.5

3.3. Desarrollo del modelo

Todo el código está recogido en un *notebook* de Jupyter en *python*, debidamente estructurado y comentado. Se han utilizado varias librerías que han facilitado el desarrollo del modelo:

- *pandas*: gestión de tablas de datos
- *numpy*: gestión de arrays n-dimensionales para estructurar los datos de manera que sirvan de input al modelo
- *sklearn*: preprocesamiento de datos
- *tensorflow*: implementación de redes neuronales de distintos tipos (densas, recurrentes, convolucionales, etc)
- *keras*: API sobre *tensorflow*, que facilita en gran medida la implementación de redes neuronales
- *math*: funciones matemáticas y constantes, que se pueden utilizar para cálculos matemáticos complejos

De todo el código, que es extenso, merece la pena destacar la función para crear el modelo, que implementa el diseño del modelo explicado en la sección anterior. En la función *create_model*, que se encuentra en el código de la siguiente hoja, se implementa una red LSTM para los valores temporales con una ventana temporal de tamaño *window_size*, a la que se le puede incluir o no una máscara para omitir los valores desconocidos con el flag booleano *mask*. Implementa en paralelo una red densa para las variables fijas, la cual es concatenada con la LSTM. Añade *hidden_layers* capas densas adicionales hasta el output. Finalmente muestra un resumen de la configuración del modelo y lo devuelve para que sea compilado, entrenado y posteriormente utilizado para realizar predicciones de la deforestación en los próximos años.

También cabe mencionar algunas librerías adicionales que se utilizan en el *notebook*, aunque no se han utilizado expresamente para desarrollar el modelo:

- *os*: gestión de directorios
- *gmplot*: librería de Google Maps [10]: mapas geográficos muy diversos (puntos dispersos, mapas de calor, líneas de cuadrícula, etc)
- *matplotlib*: grácas que muestran los resultados obtenidos, útil para analizar datos de manera visual

3.4. Data augmentation

Antes incluso de desarrollar el modelo, podemos predecir que los resultados del mismo pueden no ser muy buenos, teniendo en cuenta que se dispone únicamente de 760 municipios con datos fijos y de deforestación a lo largo de los últimos 20 años. Esto lleva a pensar que quizás se haya pocas instancias de entrenamiento para que el modelo, que es algo complejo, aprenda correctamente.

De hecho, los resultados iniciales que arroja el modelo al realizar las primeras pruebas, tienen margen de mejora. El error apenas disminuye a pesar de entrenar el modelo con 50, 100 ó 200 épocas. De esto se puede inferir que el modelo, que es algo complejo, quizás no esté siendo capaz de aprender con el volumen de datos del que se dispone.

Código 3.1: Creación del modelo: función desarrollada en *python* utilizando la librería *keras*

```
def create_model(window_size, n_vars_temp, hidden_layers, mask=1):
    '''
    Crea el modelo, que consta de una red densa para las variables fijas y una red LSTM para las
    temporales, concatenadas en una nueva densa que da el output de predicción de deforestación.
    Se ha escogido la función de activación relu, que funciona bien para regresión de valores positivos.
    El modelo se crea con una ventana temporal definida para la red LSTM y un número de capas antes
    del output, además de poder establecer una máscara para los valores nans de la LSTM.
    param window_size: tamaño de la ventana temporal para la red LSTM
    param n_vars_temp: número de variables temporales en cada paso
    param hidden_layers: número de capas desde la concatenación hasta el output
    param mask: flag para establecer o no una máscara para los valores desconocidos en la red LSTM
    returns: model
    '''
    # VARIABLES TEMPORALES
    # LSTM para variables temporales con 5 unidades de memoria y dropout para evitar sobreaprendizaje
    input_temporal = Input(shape=(window_size-1, n_vars_temp))
    if mask==0:
        lstm = LSTM(5, activation='relu', dropout=0.2)(input_temporal)
    else:
        masking = Masking(mask_value=-1)(input_temporal)
        lstm = LSTM(5, activation='relu', dropout=0.2)(masking)
    # Capa densa de 16 neuronas a continuación de la LSTM
    dense_1 = Dense(16, activation = 'relu')(lstm)

    # VARIABLES FIJAS
    # Capa densa de 32 neuronas para variables fijas
    input_fijo = Input(shape=(X2_train.shape[1],))
    dense_2 = Dense(32, activation = 'relu')(input_fijo)

    # CONCATENACIÓN DE LAS 2 REDES
    merge = concatenate([dense_1, dense_2])
    dense = [merge]
    # Capas densas adicionales de 16 neuronas previas al output
    for i in range(hidden_layers):
        dense.append(Dense(16, activation = 'relu')(dense[-1]))
    # Capa densa de 1 neurona para el output: deforestación anual
    output = Dense(1, activation = 'relu')(dense[-1])
    model = Model(inputs=[input_temporal, input_fijo], outputs=output)

    # Muestra por pantalla el resumen de las capas del modelo
    print(model.summary())

    return model
```

Como ejemplo, para un modelo entrenado con los 760 municipios durante 100 épocas, se produce overfitting a partir de la época 20 aproximadamente, lo cual se ilustra en la Figura 3.8. Esto es indicativo de que el modelo aprende conclusiones muy específicas y pierde capacidad de generalizar. Por ello, al recibir inputs nuevos con los que nunca se ha entrenado, los resultados empeoran.

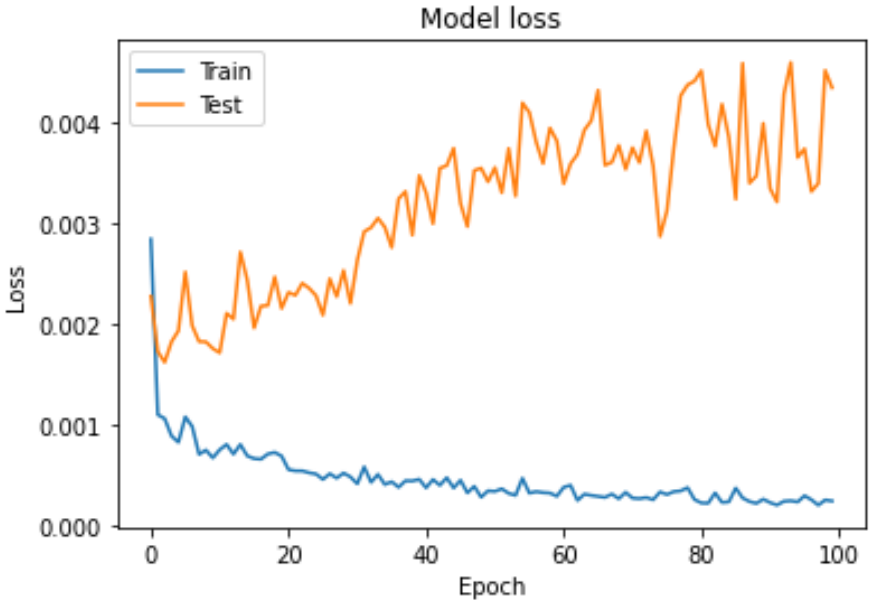


Figura 3.8: Overfitting del modelo con 760 municipios

Los resultados obtenidos en este caso no son los ideales, ya que hay un error absoluto medio de 19,15 km^2 por municipio, lo que supone un error relativo del 58,78 %. En la Figura 3.9 se pueden comparar los valores predichos con los esperados para algunos municipios.

	Municipio	Expected	Predicted	Error
0	IRACEMA (RR)	25.391160	22.221107	3.170053
1	LUCAS DO RIO VERDE (MT)	17.463458	18.509243	1.045786
2	ARIPUANÃ (MT)	129.576926	113.464562	16.112363
3	REDENÇÃO (PA)	85.615971	51.376919	34.239051
4	GONÇALVES DIAS (MA)	18.820655	20.655711	1.835056
...	...	...	...	...
755	GRAJAÚ (MA)	47.659481	35.379334	12.280147
756	SANTA TEREZA DO TOCANTINS (TO)	0.000000	27.573863	27.573863
757	CHUPINGUAIA (RO)	85.931143	59.184218	26.746925
758	PORTO ESTRELA (MT)	12.250149	29.283856	17.033707
759	IGARAPÉ GRANDE (MA)	12.217988	29.050394	16.832406

Figura 3.9: Resultados del modelo con 760 municipios

Parece lógico pensar que se pueden alcanzar mejores resultados atajando este problema. Una posible solución es generar datos falsos que repliquen la distribución de los datos reales. Esta técnica es conocida como *data augmentation*.

Con este objetivo en mente, se generan "réplicas" de los municipios con datos espurios. Estos son datos reales a los que se les ha añadido ruido, e.d. variaciones con respecto a los municipios reales, para que no sean copias exactas y el modelo sea capaz de generalizar de manera correcta. Se define el número de "réplicas" a crear para cada municipio con el parámetro *n_replics*. El ruido se establece de manera aleatoria para las siguientes variables en los siguientes rangos:

- Latitud: $\pm 10\,\%$ de la desviación típica
- Longitud: $\pm 10\,\%$ de la desviación típica
- Área total: $\pm 30\,\%$ del área total, con esto se permite una mayor variación en el área para crear una muestra con municipios pequeños y grandes
- No bosque: se mantiene la proporción con respecto al área total $\pm 10\,\%$
- Hidrografía: se mantiene la proporción con respecto al área total $\pm 10\,\%$
- Incremento deforestación: se mantiene la proporción con respecto al área total $\pm 5\,\%$, con esto se puede calcular el resto de variables temporales, asumiendo que "Nubes" y "No observado" son 0

De esta forma se crean nuevos municipios artificialmente, que se encuentran en el mismo estado (a la variable "Estado" no se le añade ruido) y están ubicados geográficamente en una zona cercana. El área puede variar más para generar municipios de tamaños variados, pero manteniendo similares las proporciones de agua y no bosque, así como la deforestación a lo largo de los años.

En la Figura 3.10 se observa la nueva distribución geográfica de los municipios para *n_replics*=10, que muestra una mayor densidad de municipios si la contrastamos con los municipios reales en la Figura 3.3. Así, se pasa de tener 760 municipios a 8360 (760 reales y 10 "réplicas" de cada uno).

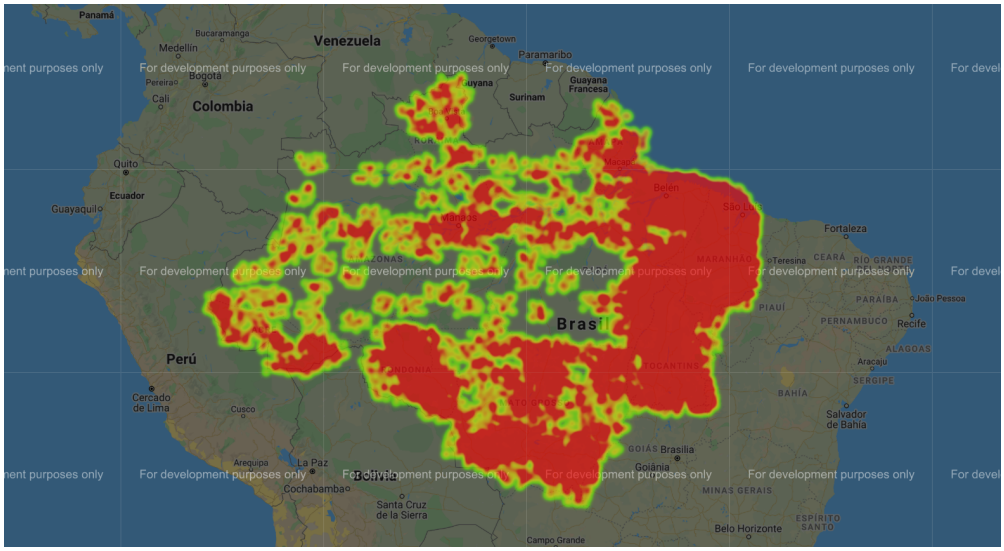


Figura 3.10: Mapa de calor de los municipios incluyendo los generados artificialmente

Al comparar los datos originales con los nuevos datos falsos, se puede concluir que tienen una distribución similar, lo cual puede obtenerse fácilmente por medio de la función *describe()* de *pandas*. Esto se refleja en la Figura 3.11.

	Latitud	Longitud	Area total	No bosque	Hidrografia	Deforestacion 2000
count	760.000000	760.000000	760.000000	760.000000	760.000000	760.000000
mean	-7.044355	-52.433004	6668.484211	1264.545921	148.586447	701.211184
std	4.777753	7.022612	13866.222990	2403.419358	430.646784	951.484350
min	-17.773150	-73.412410	64.000000	0.000000	0.000000	0.000000
25%	-10.840910	-57.289448	891.750000	4.600000	0.000000	42.950000
50%	-6.565670	-49.441275	2379.500000	195.850000	4.350000	338.850000
75%	-3.127692	-47.254840	6422.000000	1436.725000	65.025000	1001.975000
max	4.716360	-44.028960	159540.000000	19780.800000	4499.900000	7212.300000

(a) Distribución de los datos reales

	Latitud	Longitud	Area total	No bosque	Hidrografia	Deforestacion 2000
count	8360.000000	8360.000000	8360.000000	8360.000000	8360.000000	8360.000000
mean	-7.043214	-52.438101	6665.039909	1262.637220	149.636804	715.472812
std	4.780553	7.021901	14112.333587	2452.956746	440.343900	995.290239
min	-18.243955	-74.063553	50.813683	0.000000	0.000000	0.000000
25%	-10.816046	-57.327870	884.748898	4.214695	0.000000	44.116997
50%	-6.557930	-49.540860	2321.424048	194.917791	4.399782	339.322512
75%	-3.115792	-47.195251	6442.370977	1396.840110	64.231914	1006.871509
max	5.177219	-43.362929	195268.761055	26006.790192	5599.819753	9279.559240

(b) Distribución de los datos con *n_replics*=10

Figura 3.11: Comparativa de la distribución de datos reales 3.11(a) y datos artificiales 3.11(b)

3.5. Pruebas y configuración de la red

A lo largo de este capítulo se han explicado los distintos hiperparámetros que pueden ser definidos para configurar el modelo óptimo. Para realizar distintas pruebas en búsqueda del modelo que mejores resultados ofrezca, se establecen arrays que almacenan los valores de los hiperparámetros con los que se desea entrenar y testar el modelo. Se itera sobre estos arrays en un bucle for anidado para realizar todas las simulaciones. A continuación se recogen todos estos arrays:

- *n_replics*: número de réplicas de municipios que generan a partir de los municipios reales
- *hidden_layers*: número de capas intermedias de la red neuronal entre la concatenación y el output
- *epochs*: número de épocas con las que se entrenar el modelo
- *batch_sizes*: tamaño del lote de aprendizaje (número de ejemplos con los que se calcula el descenso por gradiente)
- *window_size*: tamaño de la ventana temporal para la red LSTM

Teniendo en cuenta tiempos de ejecución, se ha definido una batería de pruebas no muy costosa computacionalmente, lo que no quita que 32GB de RAM y un i7 de 6 núcleos han estado al servicio del trabajo computacional durante horas. Cabe mencionar que esto ha sido posible debido al forzoso encierro durante la cuarentena, con el ordenador de la empresa completamente disponible. Dicen que de todo lo malo hay que sacar algo bueno.

Tras realizar una serie de pruebas manuales y analizando los resultados, se han identificado unos valores en torno a los que puede encontrarse el modelo óptimo. Con ellos, se ha realizado una batería de pruebas para identificar los valores idóneos de estos hiperparámetros. Concretamente, los valores con los que se han realizado las pruebas son los siguientes:

Código 3.2: Batería de pruebas

```
1 n_replics = [0,5,10,20]
2 hidden_layers=[1,2,3]
3 epochs=[100]
4 batch_sizes=[32,64]
5 window_sizes=[4,5,6]
```

En lugar de definir varios posibles valores en *epochs*, se ha optado por entrenar con 100 épocas como límite máximo, guardando el modelo en la época en la que se han obtenido mejores resultados en validación. Para ello, se ha utilizado el callback *ModelCheckpoint* de *keras*. Además, se ha observado que en algunos casos el modelo no mejora a partir de las primeras épocas. Para evitar que este prosiga con el entrenamiento, con el coste computacional que esto conlleva, se ha definido la parada de aprendizaje si el modelo no mejora en 15 épocas consecutivas. Esto se ha implementado con *EarlyStopping* de *keras*.

Código 3.3: Entrenamiento del modelo

```
1 # checkpoint: guardamos el modelo que ha dado resultados en la mejor época
2 checkpoint = ModelCheckpoint("modelos/" + model_name, save_best_only=True, verbose=1)
3 # earlystop: si el modelo no mejora en 15 épocas consecutivas, paramos el entrenamiento
4 earlystop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)
5 # entrenamiento del modelo
6 history = model.fit([X1_train, X2_train], y_train, epochs=e, batch_size=bs, verbose=2, \
```

Se han almacenado los resultados de la batería de pruebas para todas las combinaciones de hiperparámetros. Estos nos permiten analizar cómo se comporta el modelo en función de los distintos hiperparámetros. Se pueden observar en la Tabla 3.1.

Tabla 3.1: Resultados de la batería de pruebas

Municipios	Window size	Capas intermedias	Batch size	Error absoluto medio
760	4	1	32	20.74718
760	4	1	64	17.56554
760	4	2	32	18.16932
760	4	2	64	19.77147
760	4	3	32	18.43061
760	4	3	64	19.33909
760	5	1	32	16.08591
760	5	1	64	14.8799
760	5	2	32	14.76014
760	5	2	64	32.68424
760	5	3	32	14.60926
760	5	3	64	32.68424
760	6	1	32	17.34971
760	6	1	64	16.34617
760	6	2	32	14.46266
760	6	2	64	23.65765
760	6	3	32	16.97257
760	6	3	64	19.05864
4560	4	1	32	12.03387
4560	4	1	64	16.65401
4560	4	2	32	15.27716
4560	4	2	64	20.31839
4560	4	3	32	17.4646
4560	4	3	64	33.64384
4560	5	1	32	17.04853
4560	5	1	64	18.45091
4560	5	2	32	16.87893
4560	5	2	64	14.84898
4560	5	3	32	17.05211
4560	5	3	64	18.18842
4560	6	1	32	23.6218
4560	6	1	64	15.31802
4560	6	2	32	17.58198
4560	6	2	64	11.85884

Continued on next page

Tabla 3.1 – continued from previous page

Municipios	Window size	Capas intermedias	Batch size	Error absoluto medio
4560	6	3	32	33.82193
4560	6	3	64	18.01594
8360	4	1	32	14.11515
8360	4	1	64	19.83043
8360	4	2	32	15.39117
8360	4	2	64	13.97293
8360	4	3	32	15.01827
8360	4	3	64	34.20584
8360	5	1	32	16.41191
8360	5	1	64	16.55947
8360	5	2	32	34.31491
8360	5	2	64	16.91641
8360	5	3	32	13.49329
8360	5	3	64	15.10105
8360	6	1	32	15.23977
8360	6	1	64	20.52418
8360	6	2	32	16.89536
8360	6	2	64	15.81259
8360	6	3	32	19.43007
8360	6	3	64	16.43419
15960	4	1	32	32.14322
15960	4	1	64	12.61125
15960	4	2	32	14.64619
15960	4	2	64	18.26267
15960	4	3	32	32.14322
15960	4	3	64	22.16912
15960	5	1	32	14.17771
15960	5	1	64	14.91879
15960	5	2	32	11.62285
15960	5	2	64	12.44124
15960	5	3	32	32.25768
15960	5	3	64	16.29228
15960	6	1	32	13.35936
15960	6	1	64	18.09059
15960	6	2	32	14.93252

Continued on next page

Tabla 3.1 – continued from previous page

Municipios	Window size	Capas intermedias	Batch size	Error absoluto medio
15960	6	2	64	17.50259
15960	6	3	32	32.32595
15960	6	3	64	14.52978

Los resultados de esta batería de pruebas muestran que el modelo óptimo encontrado ha aprendido de 15960 municipios (*n_replics*=20), ventana temporal de tamaño 5, 2 capas intermedias y un batch size de 32 instancias para calcular el descenso por gradiente. Su error absoluto medio al calcular el incremento de deforestación por municipio es de 11.62 km^2.

Un detalle importante a mencionar es que el error disminuye gracias al *data augmentation*. Agregando lo resultados por número de municipios con los que se ha entrenado el modelo, se puede observar que el error mínimo es menor para 4560 (*n_replics*=5), 8360 (*n_replics*=10) y 15960 (*n_replics*=20). Esto se aprecia en la Figure 3.12

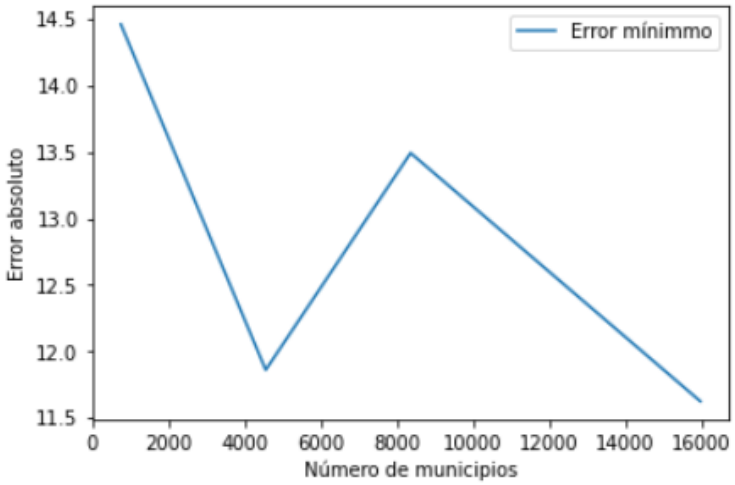


Figura 3.12: Comparativa de errores para distintos números de municipios

4

Resultados, conclusiones y trabajo futuro

Una vez escogido el modelo óptimo, este se puede utilizar para realizar la predicción de la deforestación en los municipios de la Amazonia en los próximos años. Agregando los resultados de los municipios, se puede inferir el futuro que le depara a la selva amazónica. En este capítulo, además de los resultados, se desarrollan una serie de conclusiones y se establece una posible continuación del trabajo para mejorar el modelo y seguir estudiando la deforestación en el bosque más grande del planeta.

4.1. Resultados obtenidos

Gracias a la batería de pruebas realizada se ha logrado identificar una buena configuración para el modelo. Se ha demostrado empíricamente que, ante la falta de datos, el *data augmentation* puede ser de gran ayuda para el correcto aprendizaje del modelo. Así, se ha identificado que la configuración óptima del modelo es:

- Número de “réplicas” de municipios: 20
- Capas intermedias: 2
- Batch size (número de ejemplos con los que se calcula el descenso por gradiente): 32
- Tamaño de la ventana temporal: 5

Su función de error dista bastante del overfitting de la Figura 3.8, con un modelo entrenado únicamente con 760 municipios. En la curva de error de este modelo, mostrado en la Figura 4.1, se puede apreciar un continuo aprendizaje hasta la etapa 31. A partir de ahí parece que el error comienza a aumentar.

El modelo tiene un error absoluto medio de 11,62, lo que supone un error relativo del 36,03 %. Teniendo en cuenta que en algunos municipios se estima una predicción mayor que la real y en otros una menor, si se compara la deforestación agregada, el error es del 17,36 %. Por esto, se concluye que se puede utilizar el modelo para estimar la deforestación en los 760 municipios y analizar las cifras de manera agregada.

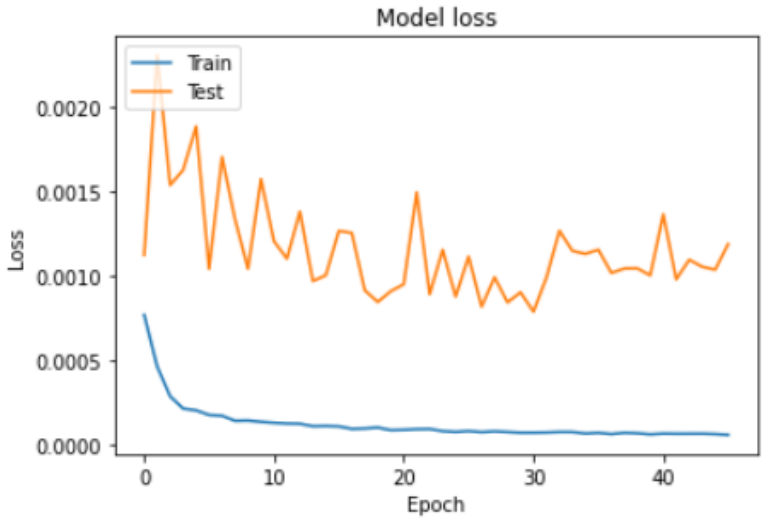


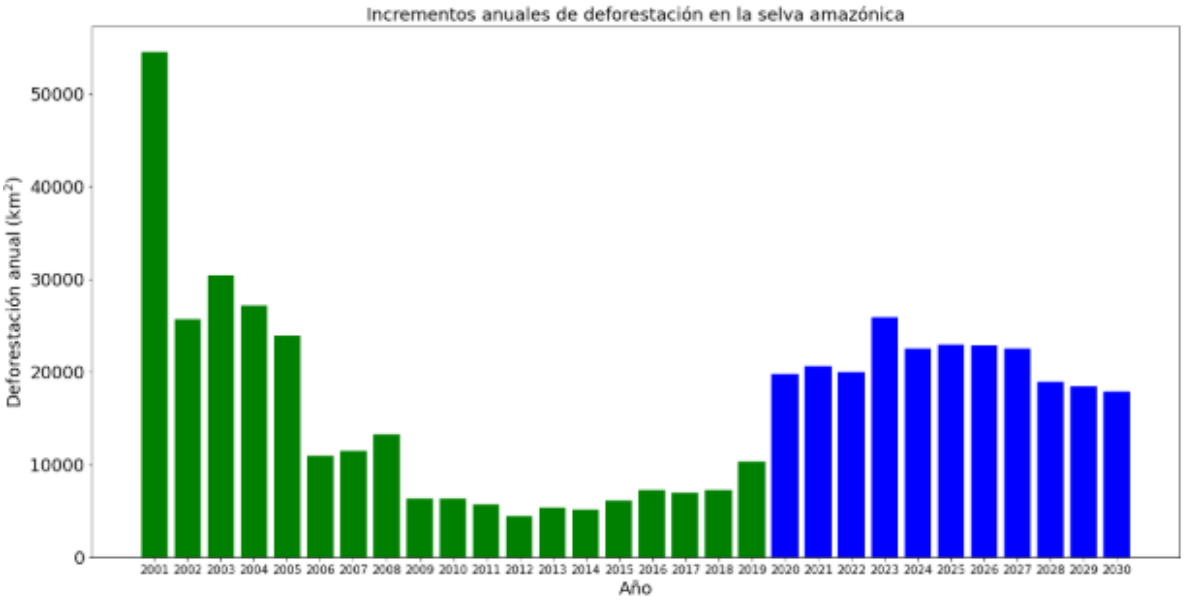
Figura 4.1: Aprendizaje del modelo definitivo

Para obtener las predicciones de cada municipio en años futuros, se recogen los datos de los *window_size*-1 años más recientes y se predice la deforestación del siguiente año. Una vez se dispone de esta, para generar la predicción de la deforestación en el siguiente año se utilizan nuevamente los *window_size*-1 años anteriores, incluyendo la predicción generada. Esto se repite sucesivamente, utilizando las predicciones que se van generando como input para realizar la siguiente predicción. Cabe destacar que, cuantos más años se desee predecir, más inexactos serán los resultados, principalmente por 2 razones:

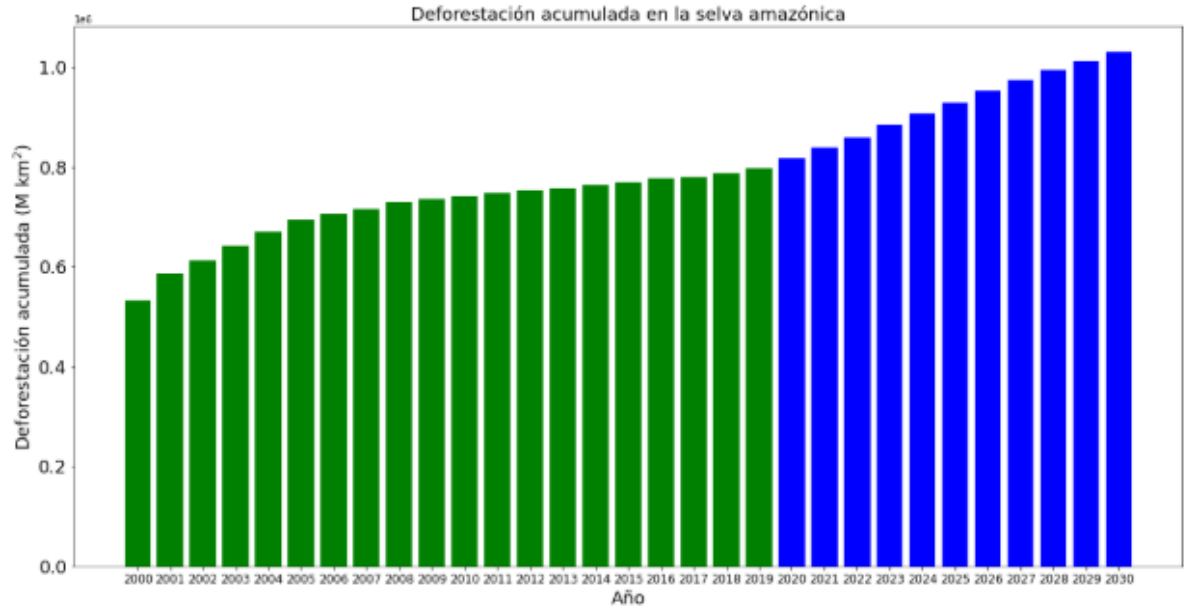
- Cuantos más datos predichos y menos datos reales se utilizan para generar una nueva predicción, más sesgo hay en la predicción.
- A medida que se trata de predecir la deforestación en un año más lejano, mayor es la incertidumbre, ya que el contexto político, social y económico puede variar en un sentido más o menos beneficioso para la protección de la Amazonia.

Se ha realizado la predicción de la deforestación de los 760 municipios hasta 2030 y se han sumado los datos para obtener las cifras agregadas. En la Figura 4.2 se exponen los datos de deforestación anuales, incluyendo los datos hasta la fecha en verde y los predichos en azul.

A la luz de los datos reales, se puede observar que, tras unos años en los que se ha logrado disminuir la deforestación, se está volviendo a una tendencia alcista. El modelo predice una continuidad de esta tendencia, pero a un ritmo notablemente mayor. Quizás se haya sobreestimado la deforestación en los próximos años, quizás no. Solo el tiempo lo dirá.



(a) Incrementos anuales de deforestación en la selva amazónica



(b) Deforestación acumulada en la selva amazónica

Figura 4.2: Deforestación en la Amazonia entre 2000 y 2030

4.2. Conclusiones

Si bien el modelo no tiene la precisión idónea, este puede ser de utilidad para identificar el pronóstico aproximado de deforestación en los próximos años. Tras analizar los resultados obtenidos, queda claro que la situación no es la ideal y se debe revertir la tendencia para dar un futuro esperanzador a la Amazonia. Consuela pensar que la selva amazónica abarca entre 5,5M y 6,7M km^2 y desde 2008 a 2018 se ha deforestado menos de 8.000 km^2 anuales, cifras bastante pequeñas. En cualquier caso, si no se llevan a cabo planes de reforestación, la Amazonia va a seguir disminuyendo. Lo más preocupante es que en 2019 la deforestación ha superado los 10.000 km^2, lo cual ha podido influir en que el modelo prediga esta tendencia alcista. Se estima que para 2030 la deforestación acumulada será de 1M km^2.

El contexto socioeconómico de Brasil, agravado por la pandemia global, que actualmente es el centro de atención, puede no ser el más favorable para tomar las medidas pertinentes para proteger la Amazonia. En cualquier caso, cuando esto pase, que pasará, se debe prestar atención al problema de la deforestación. Todavía no es tarde para salvar la Amazonia, pero se debe empezar a trabajar en definir un plan de acción que tenga impacto en el medio y largo plazo.

El ser humano lleva siglos utilizando los recursos que ofrece el planeta para avanzar y mejorar la calidad de vida. En el pasado, era desconocido el impacto que tenían nuestras acciones en el medioambiente. A día de hoy, esto no puede servir de excusa. Conceptos como cambio climático, sostenibilidad o energías renovables no nos son ajenos. Debemos ser conscientes de que el planeta en el que vivimos nos ha sido prestado durante nuestra estancia en él. Debemos cuidarlo para dejarlo en igual o mejor estado que lo hemos recibido. De este modo, las generaciones venideras podrán disfrutar de él como hoy hacemos nosotros.

4.3. Trabajo futuro

Este Trabajo de Fin de Grado asienta las bases de un modelo de predicción de deforestación de municipios de la Amazonia, combinando redes neuronales densas y LSTM. Durante los años venideros se pueden contrastar los resultados de las predicciones y realizar reentrenamientos para actualizar el modelo con nuevos datos. Incluso si el INPE pone a disposición del público nuevas variables que puedan ser utilizadas, se puede enriquecer el modelo. De igual manera, también se pueden cruzar los datos con otras fuentes que incluyan variables socioeconómicas de los 760 municipios o de los estados a los que pertenecen los mismos.

Otra posible vía de actuación es desarrollar un nuevo modelo con otra tipología de datos. Sería interesante disponer de imágenes de la Amazonia para identificar de manera precisa la deforestación anual. Una red neuronal convolucional podría servir para diferenciar la zona deforestada del bosque.

Bibliografía

[1] Greenpeace, “Brazil and the amazon forest,” 2018. (Descargar).

[2] A. Geron, *Hands-on Machine Learning with Scikit-Learn and Tensorflow*. O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, second ed., June 2019.

[3] S. R. y Cajal. Dibujo de una lámina del córtex cerebral.

[4] G. Sanderson, “Gradient descent, how neural networks learn. deep learning, chapter 2.” 3Blue1Brown, canal docente de Youtube.

[5] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” *Cornell University*, no. E3, 2015. (Descargar).

[6] F. S. Fernández, *Propiedades del gradiente*. Universidad de Extremadura, Facultad de Ciencias, Departamento de Matemáticas.

[7] G. Sanderson, “Gradient descent, how neural networks learn. deep learning, chapter 3.” 3Blue1Brown, canal docente de Youtube.

[8] C. Olah, “Understanding lstm networks,” *Colah’s blog*, 2015. (Descargar).

[9] I. N. de Pesquisas Espaciais, “Incrementos de desmatamento acumulado - amazônia - municípios.” (Descargar).

[10] G. Maps, “gmplot.” (Documentación).


UAM
UNIVERSIDAD AUTONOMA
DE MADRID